NASA GSFC Advanced Mission Application Branch (Code 583)

ASSET SOFTWARE DESIGN DOCUMENT

May 13, 2003

Software Design Document for ASSET

Prepared by:
Jeffrey M. Robinson
Aquilent, Inc.

Document Summary

Document Title	Software Design Document for SPLAT
Author	Jeffrey Robinson
Status	Draft

Document Change History Log

Date of Change	Version	Summary of Change
April 23, 2003	1.0	Initial Release

Approvals

Title	Name	Signature	Date

Table of Contents

1 Introduction

The purpose of this document is to describe the software architecture for the Advanced Ssr SchEduling Tool (ASSET), formerly SSR Playback Automation Tool (SPLAT) and Goal Oriented Commanding (GOC) developed at the NASA Goddard Space Flight Center Advanced Mission Automation Branch (Code 583). The document describes the architecture for the ASSET tool and includes all changes made during the Phase I of the FY03 modifications. It will be used to drive the design and implementation of the system.

2 References

The following references were used in preparation of this document:

1. SPLAT System Requirements Specification version 1.2 April 18, 2002.
2. SPLAT Software Architecture Document version 1.1 May 28, 2002.
3. ASSET Delta System Requirements Specification
4. ASSET Delta Software Architecture Document

3 Architectural Representation

This document presents the system architecture as a series of views: A Logical View and a Process View. These views are presented as Together Models that use the Unified Modeling Language (UML).

The Logical View of the architecture describes the most important classes. Class diagrams are included to illustrate the relationships between architecturally significant classes and other elements within the system

The Process View describes the tasks (processes and threads) involved in the system's execution, their interactions and configurations. It also describes the allocation of objects and classes to tasks.

4 Architectural Goals & Constraints

This section enumerates key requirements and system constraints that have a significant bearing on the architecture. They are:

1. The architecture should support retrieving report data either from the MMS system automatically or via manual placement of input files in a common directory on the operator's local machine (PC).
2. The Report Manager function, when retrieving files from the MMS system, assumes that file names for the required reports can be distinguished using a combination of MMS file naming conventions and file content.
3. The primary user interfaces for playback scheduling, etc. must be able to run on a user's Java enabled PC.
4. The Report Retrieval must be able to secure FTP request files to the MMS UNIX workstation.
5. The Report Retrieval mechanism must bet able to retrieve the MMS generated reports from an accessible file server.
6. Data transfer (FTP) connections to the MMS workstation and the file server must be secure and run through secure shell (SSH).
7. All usability, reliability, performance and loading requirements as stipulated in the SPLAT System Requirements Document [1], must be taken into consideration as the architecture is being developed.

5 Logical View

In the Logical View the collaborations detailed in the Use Case View are combined into single class diagram that depicts the significant architectural elements. These are then organized into packages and service layers to create an Architectural Overview. The results of an analysis of concurrency and interprocess communication requirements are presented in the Process View.

5.1 Packages and Subsystems

The ASSET system is organized into four layers. The User Services Layer provides user interfaces for workflow management and interaction with schedules. The Scheduling Services Layer encapsulates the processing required to ingest and translate MMS scheduling reports, and create SSR buffer playback schedules, the Data Management Layer provide services for managing schedules and the information required to generated schedules, and the support services layer which contains utilities and tools used throughout the tool.

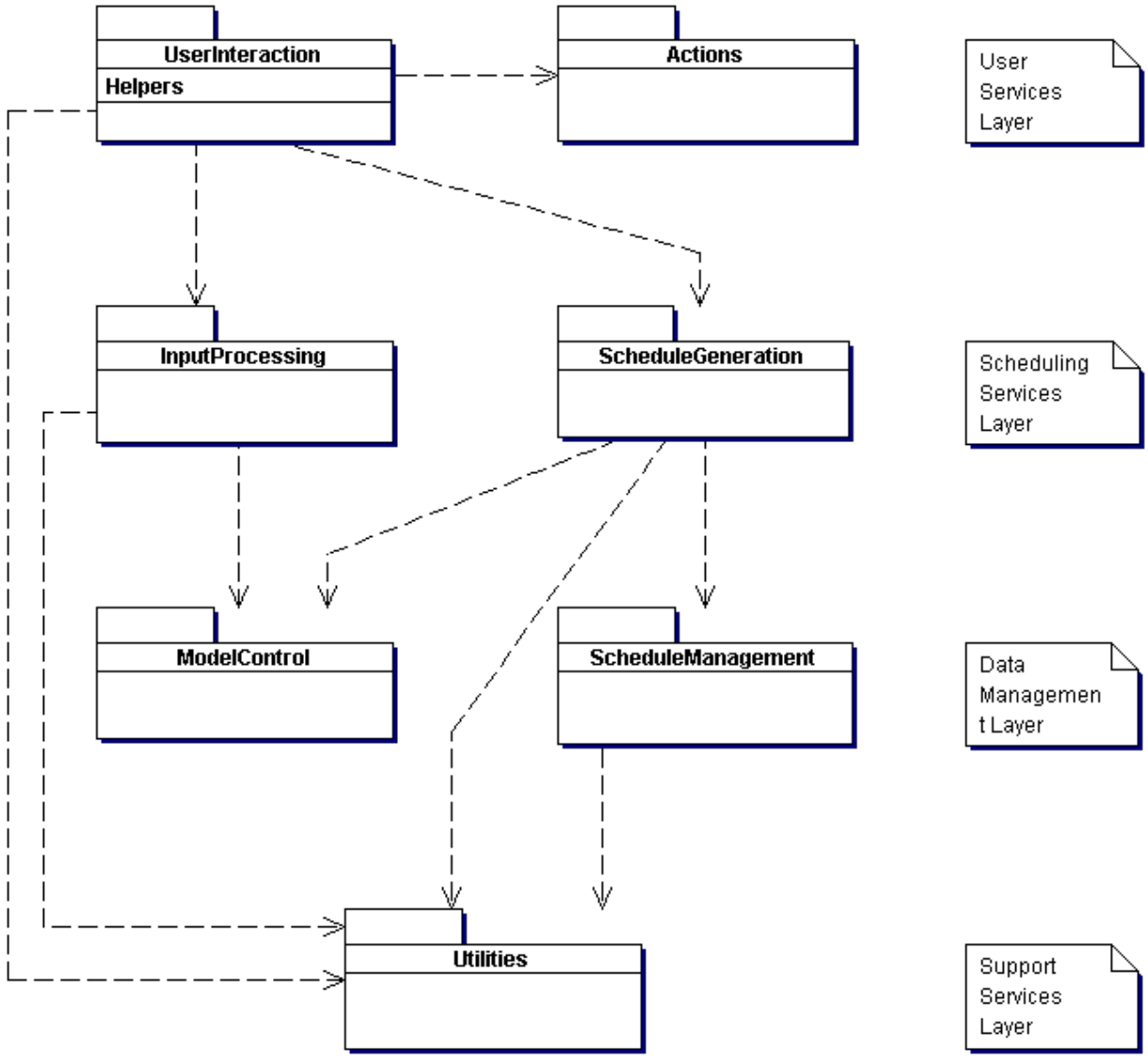


Figure 5–1: Packages and Layering for ASSET

5.1.1 User Interaction

The User Interaction package contains the GUI components and supporting classes (helpers) that allow the user to enter special event scheduling windows, select print and display filters, view the playback scheduler, and control schedule generation. These include:

- Main Window

- Print Schedule UI
- Save Schedule UI
- Parameter UI
- Filter UI
- Dump Windows UI
- Sync Point Parameters UI
- Scheduling Options UI
- Red & Yellow Limits UI
- Stations UI
- Dump Window Offsets UI
- Input Reports UI

5.1.2 Actions

The Actions package contains classes and components that provide an interface between the GUI and the underlying functionality of the system. In a sense these classes provide the glue and connection between the ASSET user interface components and the backend of the system.

5.1.3 Input Processing

The Input Processing package contains the classes and interfaces required to ingest and manage the input report files from the MMS or a local directory, and extract the contacts and MODIS/MISR mode change events from them. These include:

- Report Manager
- Event Contact Manager
- Report List
- Buffer States
- MMS Interface
- Playback Windows

5.1.4 Schedule Generation

The Schedule Generation package contains the classes that control and execute creation of SSR buffer playback schedules. The included classes are:

- Schedule Manager
- Dump Windows
- Sync Point Parameters

5.1.5 Model Control

The Model Control package contains classes that provide information describing modeling parameters and modes of operation for the Terra instruments as well as scheduling options affecting operation of the tool. These include:

- Station Status
- Dump Window Offsets
- Red & Yellow Limits
- Modeling Parameters
- Scheduling Options

5.1.6 Schedule Management

The Schedule Management package contains classes that provide control over schedule storage and display. These include:

- Event Filters
- Print Display Filters
- Playback Schedule

5.2 Design Model

This section describes in greater detail the classes identified as architecturally significant in the Software Architecture Document. The classes are documented through a series of class diagrams describing the classes themselves, and a series of sequence diagrams depicting interactions between the architecturally significant classes and other elements in the system such as User Interfaces.

5.2.1 Class Diagrams

This section contains class diagrams for the new and or modified classes in ASSET and descriptions of the major methods for each class. Note that not all methods are mentioned here. For the sake of brevity, descriptions for get and set methods are omitted.

Note that persistent data storage for the ASSET system is being provided by the Java Application Shell (JAS). Class diagrams are not provided for JAS classes.

5.2.1.1 Buffer States

The Buffer States class contains information for a single buffer state entry. This information represents a Buffer State entry extracted from the SSR Buffer States Report. Each Buffer State entry contains the current usages and durations for the SSR's ASTER, MISR and MODIS buffers. The three entries together make up the contents a Buffer State entry and represent the fullness of each of the SSR buffers at a given time. The Buffer States are used during scheduling to determine the Synchronization Point.

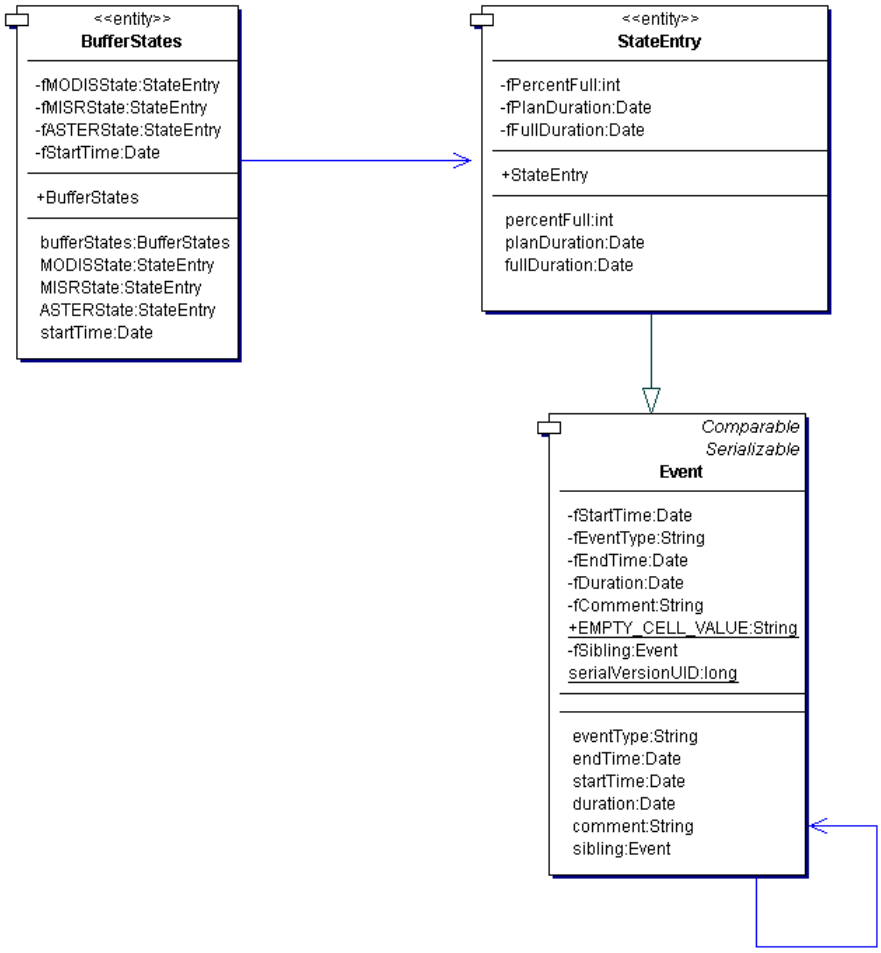


Figure 5-1: BufferStates Class Diagram

5.2.1.1.1 Major Methods

None

5.2.1.2 Contact Window

The Contact Window class contains information corresponding to an individual contact period. Contact Periods are areas of opportunity for dumping the contents of the SSR Buffers. For the ASSET system, Contact Window entries represent K and S band contacts extracted from the TDRS Contact Report and X band contacts extracted from the Ground Network (GN) Report.

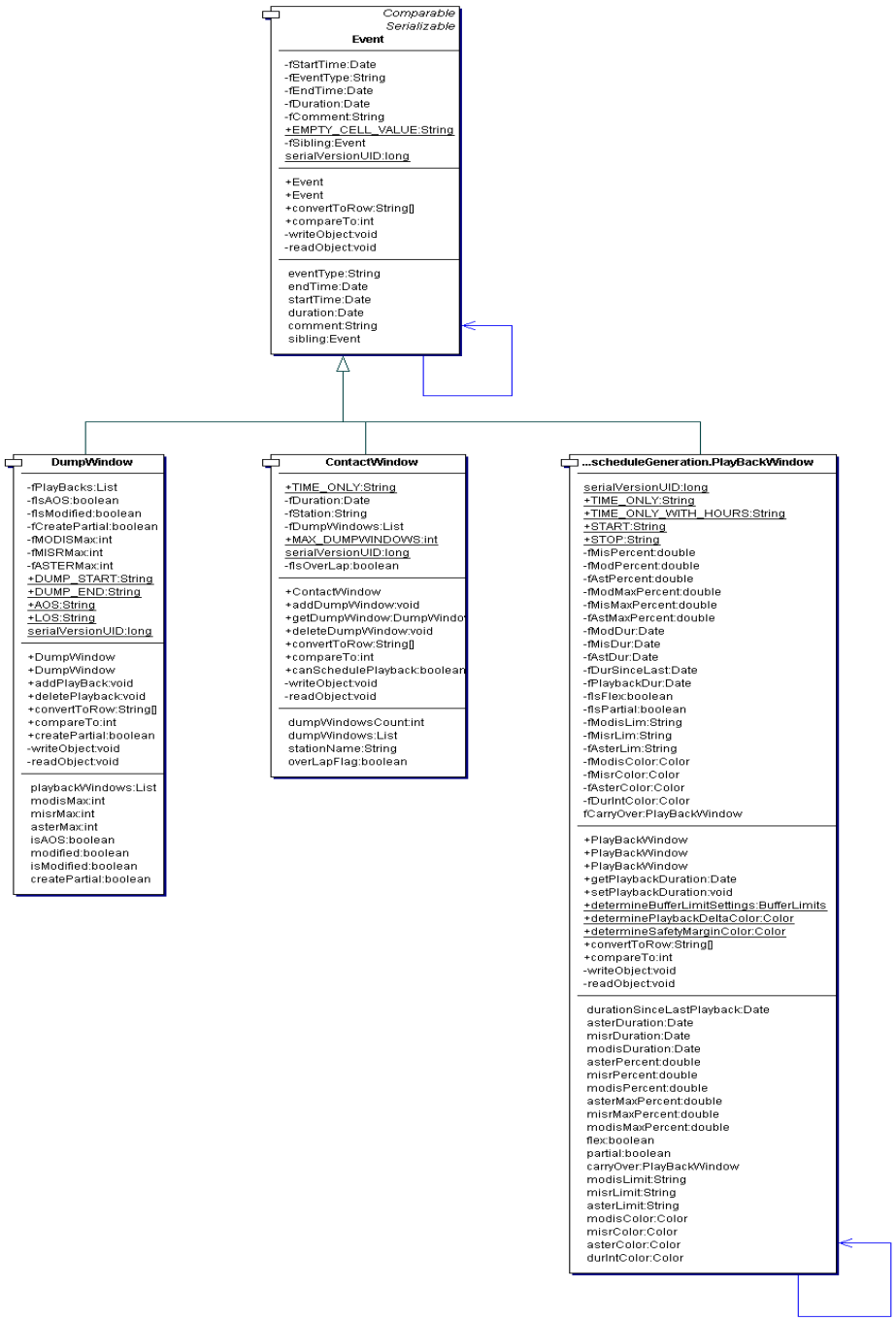


Figure 5-2: ContactWindow class diagram

5.2.1.2.1 Major Methods

- ***public int getDumpWindowCount()*** – returns the number of dump windows associated with the contact period.
- ***public int addDumpWindow(Dumpwindow dump)*** – adds the specified dump window to the contact period.
- ***public int deleteDumpWindow(DumpWindow dump)*** – removes the specified Dump Window from the contact period.
- ***Public String[] convertToRow()*** – converts the elements of the Contact Window object to an Array of String Objects.

5.2.1.3 Dump Windows

The Dump Window class represents the values describing a single dump window. Dump windows represent areas of opportunity within a contact period for playback of the SSR buffers. Each Contact Period contains at most two dump windows per contact. Each dump window can have an unlimited number of playback windows. Note that the details of the playback window class are not described in this diagram, nor is the relationship with the ContactWindow class. The intent here was to represent the relationship between a Dump Window and it's associated Playback Windows. For more information regarding the Contact Window class refer to Section 5.2.1. For more information on the Playback Window class pleaser refer to Section 5.2.1.9.

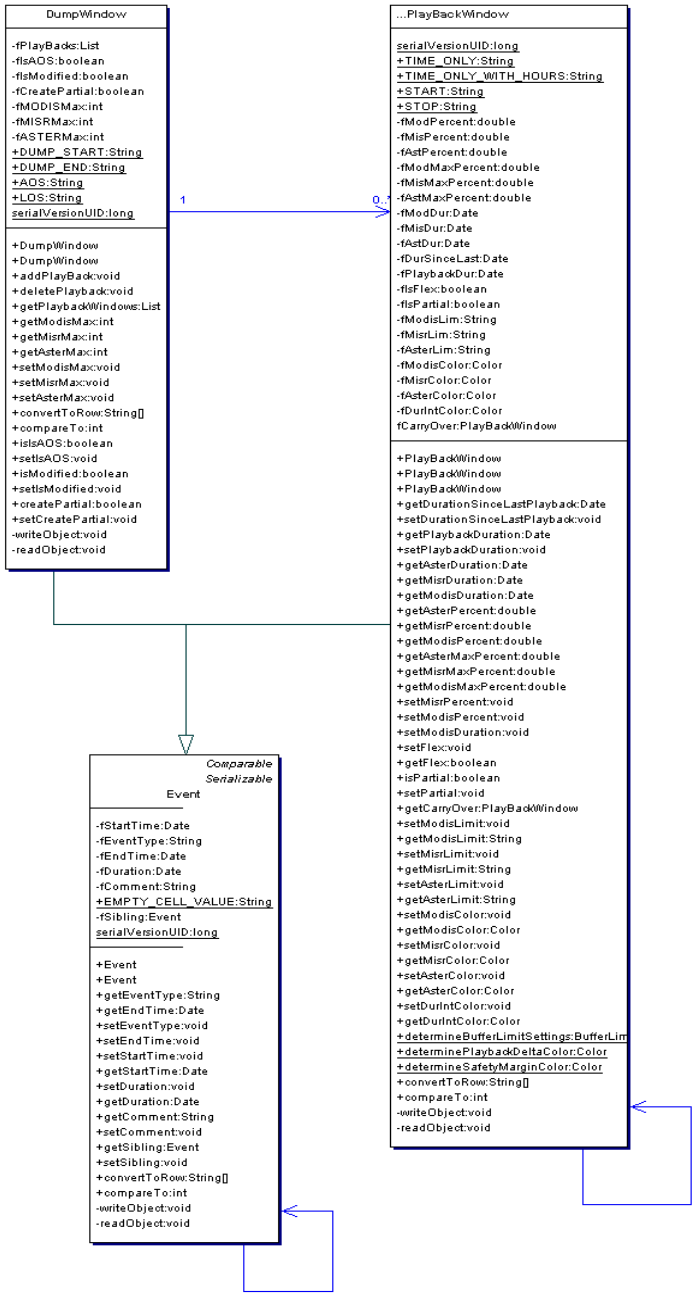


Figure 5-3: DumpWindows class diagram

5.2.1.3.1 Major Methods

- ***public void addPlayback(PlaybackWindow playback)*** – adds the specified playback window to the list of playbacks for this DumpWindow.
- ***public void deletePlayback(PlaybackWindow playback)*** – deletes the specified playback window from the current dump window.
- ***public List getPlaybackWindows()*** – returns a List of playback windows for the dump window.
- ***public String[] convertToRow()*** – returns the contents of this dump window formatted for display on the timeline.
- ***public int compareTo(Object obj)*** – dump window implementation of the compareTo operation. Allows comparison of the start time of this dump window vs. the start time of another.
- ***public boolean createPartial()*** – returns true if partial 4000 playbacks can be created in this dump window.

5.2.1.4 Event Contact Manager

The Event Contact Manager is a data repository for whose main responsibility is managing a list of extracted events composed of contact periods, MODIS and MISR mode change events, ASTER modes, and buffer states extracted from the TDRS Contact report, GN Report, SSR Buffer states report, and ATC Load Report. Note that the Event class and all classes that derive from it are shown in this diagram. The intent here is to show which types of Events are maintained in the EventContactManager.

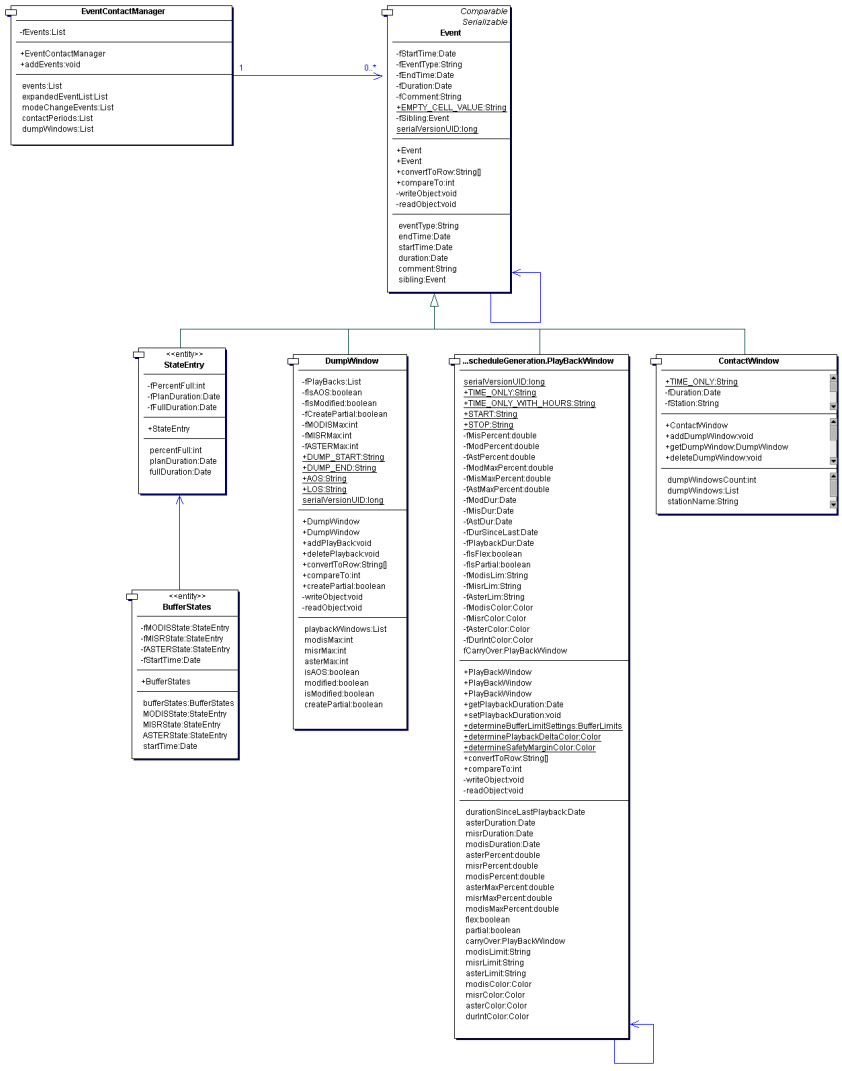


Figure 5-4: EventContactManager class diagram

5.2.1.4.1 Major Methods

- ***public void addEvents(List events)*** – add the provided list of events to the EventContactManager's list of parsed events.

5.2.1.5 Input Reports

The ASSET Tool requires a series of input reports for schedule generation. The reports required for schedule generation contain data such as orbital events, contact periods, and buffer states. Separate report classes are maintained for each report type. All reports all extend from the common InputReports base class and contain a parse routine that provides parsing services using the JAVA regular expression classes.

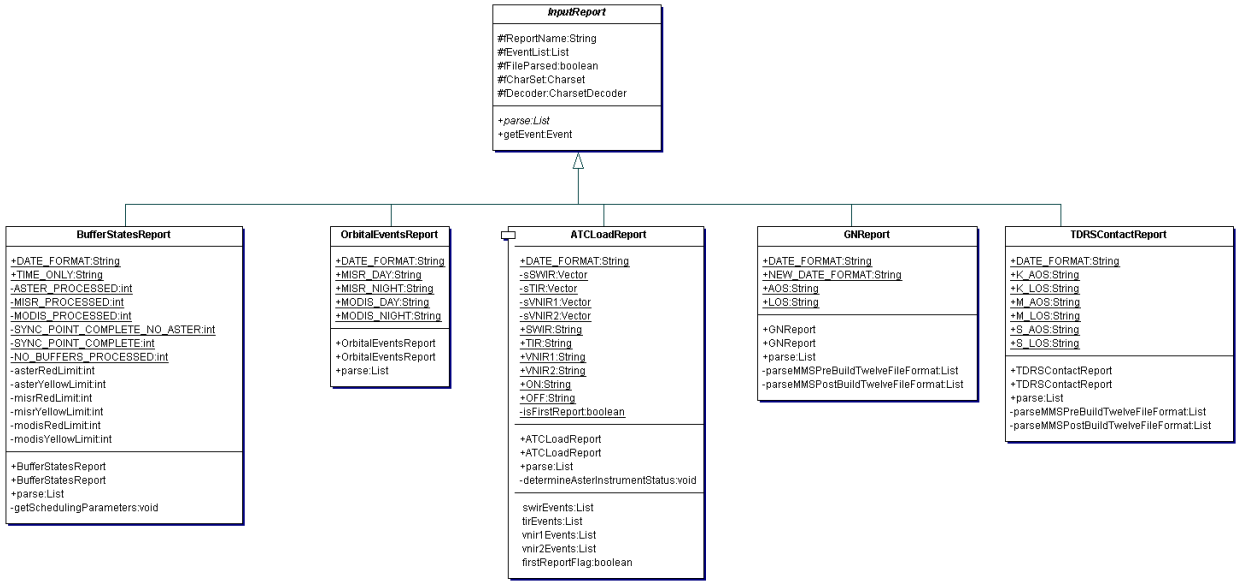


Figure 5-5: InputReports class diagram

5.2.1.5.1 Major Methods

- ***public List parse()*** – This abstract method is overridden in the child class and contains code to parse the particular report type.

5.2.1.6 MMS Report Retriever

The MMS Report Retriever class represents the ASSET interface to the Mission Management Software (MMS) system. This class resides with the other input processing classes and provides secure shell (SSH) services for connection to the MMS system for report generation requests, and to a file server for report retrieval. All connections to the MMS system and the file server will be made via secure FTP.

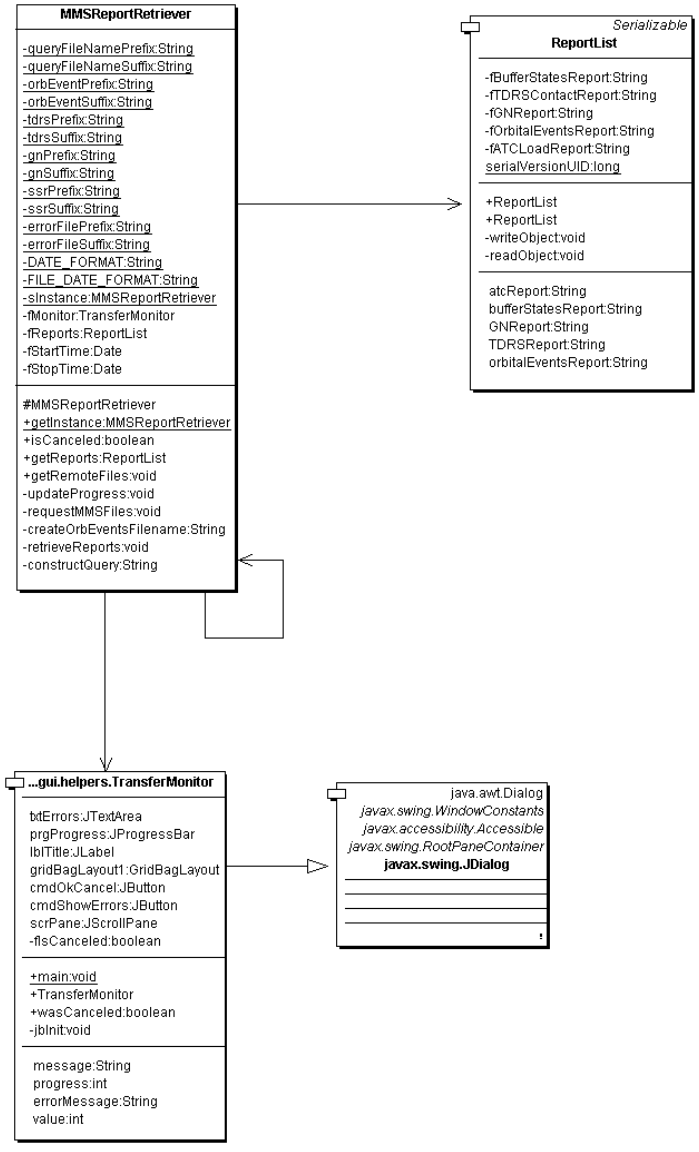


Figure 5-6: MMSReportRetriever class diagram

5.2.1.6.1 Major Methods

- ***public void updateProgress(final String msg, final String errMsg, final int value)*** – updates the status bar and messages in the status dialog.
- ***public void requestMMSFiles()*** – creates and transfers a request file to MMS for the required reports.
- ***public void retrieveReports()*** – retrieves the MMS generated reports from the File Server.
- ***public void constructQuery()*** –creates the call sequence for the MMS database extraction request.

5.2.1.7 Modeling Parameters

The Modeling Parameters are composed of those values used during schedule determination that affect schedule generation. These values include imaging rates for the individual instruments (ASTER, MISR, MODIS), playback rates for TDRS (k-band) and Ground (x-band contacts), ASTER imaging modes, buffer capacities and the alike. These values are persistent entities in the SPLAT Tool and will remain from SPLAT session to session. The Modeling Parameters are being handled by the Java Application Shell (JAS) constructs and thus are not modeled with a class diagram. For more information regarding the modeling parameters, please refer to the Section 0.

5.2.1.8 Playback Schedule

The Playback Schedule class represents the completed SSR playback schedule. It contains all contacts, dump windows, playback windows, buffer states, mode changes, and comments generated and/or added to a schedule. Note that the values stored in this class represent the full and unfiltered schedule. Hardcopy schedules and the data displayed on the MainWindow timeline are based on the data in this class.

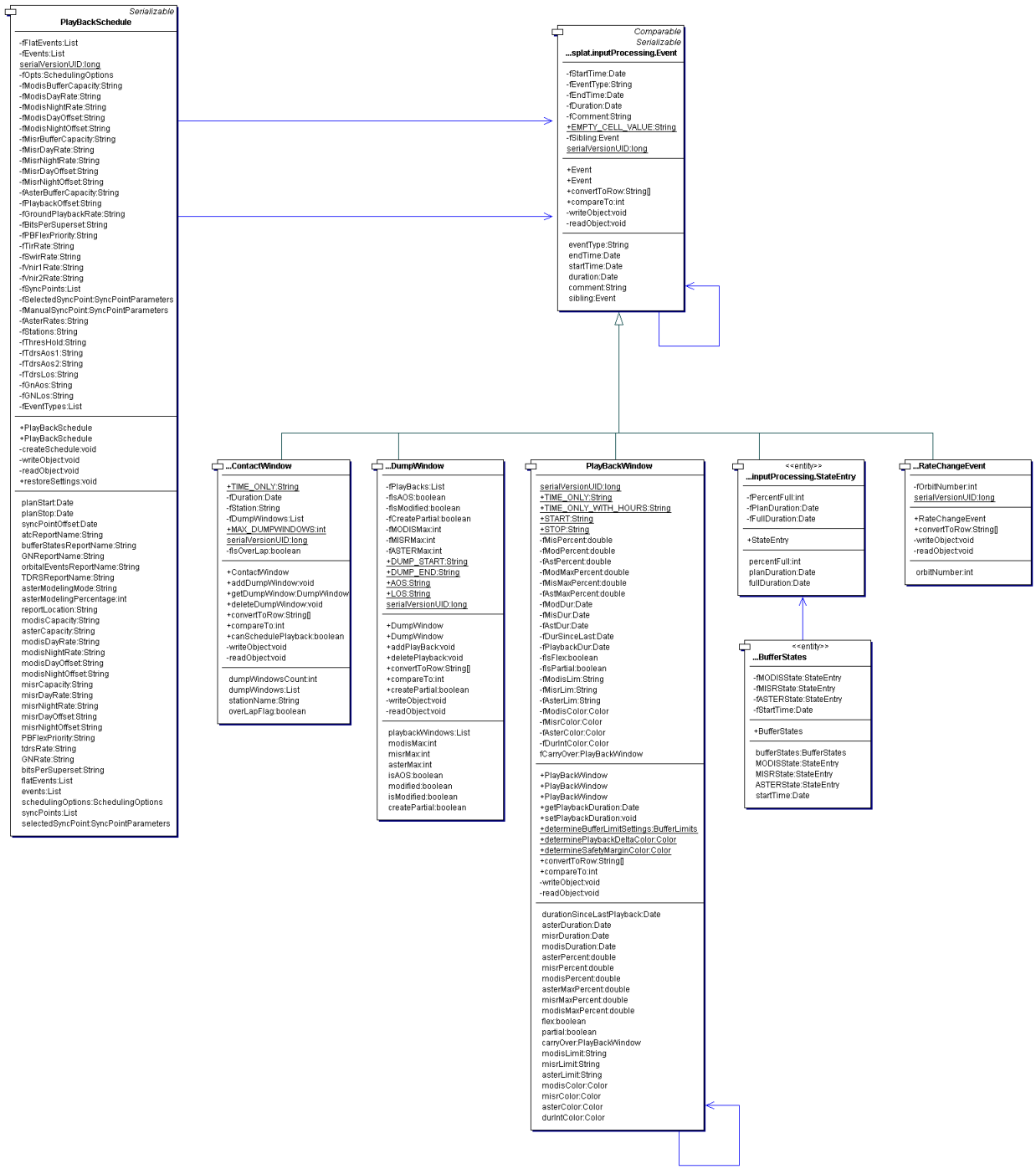


Figure 5-7: PlayBackSchedule class diagram

5.2.1.8.1 Major Methods

- ***public void createSchedule()*** – creates the playback schedule for display and hardcopy.
- ***public void restoreSettings(MainWindow main)*** – restores the ASSET system settings when a playback schedule is reloaded.

5.2.1.9 Playback Windows

The PlayBackWindow class contains all data necessary to represent a single playback window for the SSR buffers. Playback windows are created during schedule generation by the Schedule Manager control class. The placement and settings of an individual playback window are determined based on the dump windows locations, mode changes, and buffer usage.

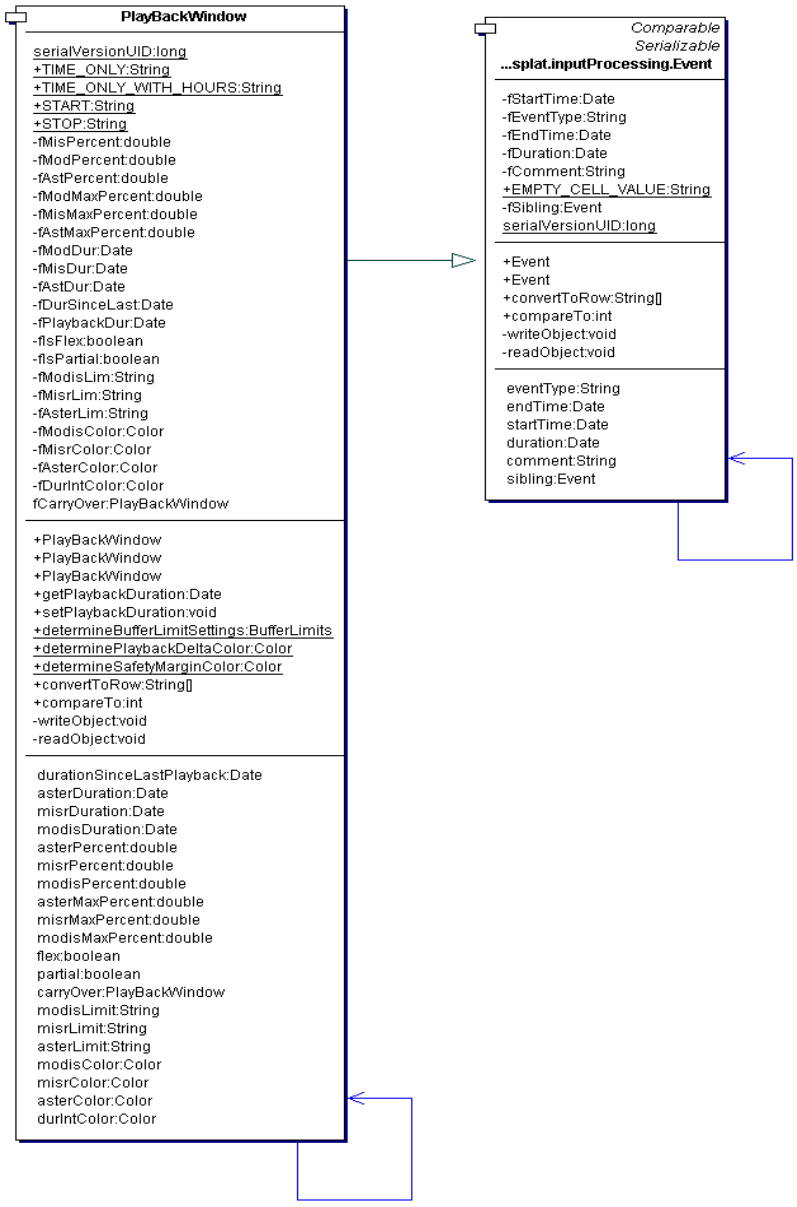


Figure 5-8: PlayBackWindow class diagram

5.2.1.9.1 Major Methods

- ***public BufferLimits determineBufferLimitSettings(double usage, int yellowLimit, int redLimit)*** – determines the buffer limit settings for a specified buffer.
- ***public Color determinePlaybackDeltaColor(long value, long yellowLimit, long redLimit)*** – determines the appropriate color for the playback delta field.
- ***public Color determineSafetyMarginColor***(***long value, long yellowLimit, long redLimit***) – determines the appropriate color for the playback safety margin field.

5.2.1.10 Print Display Filters

The Print and Display Filters classes contain information regarding the printed and/or visible fields and event types. Through these classes, the ASSET customizes the display of data on the timeline and in hardcopy output. Note that the Print Filters and Display Filters classes are separate instantiations of the Filter class presented here. The Filter class contains entries for both event filters and field filters.

Note that the list of all possible filterable fields in ASSET is static and will not change. However, the list of filterable event types is dynamic and will change from ASSET session to session depending on which reports are used for schedule generation.

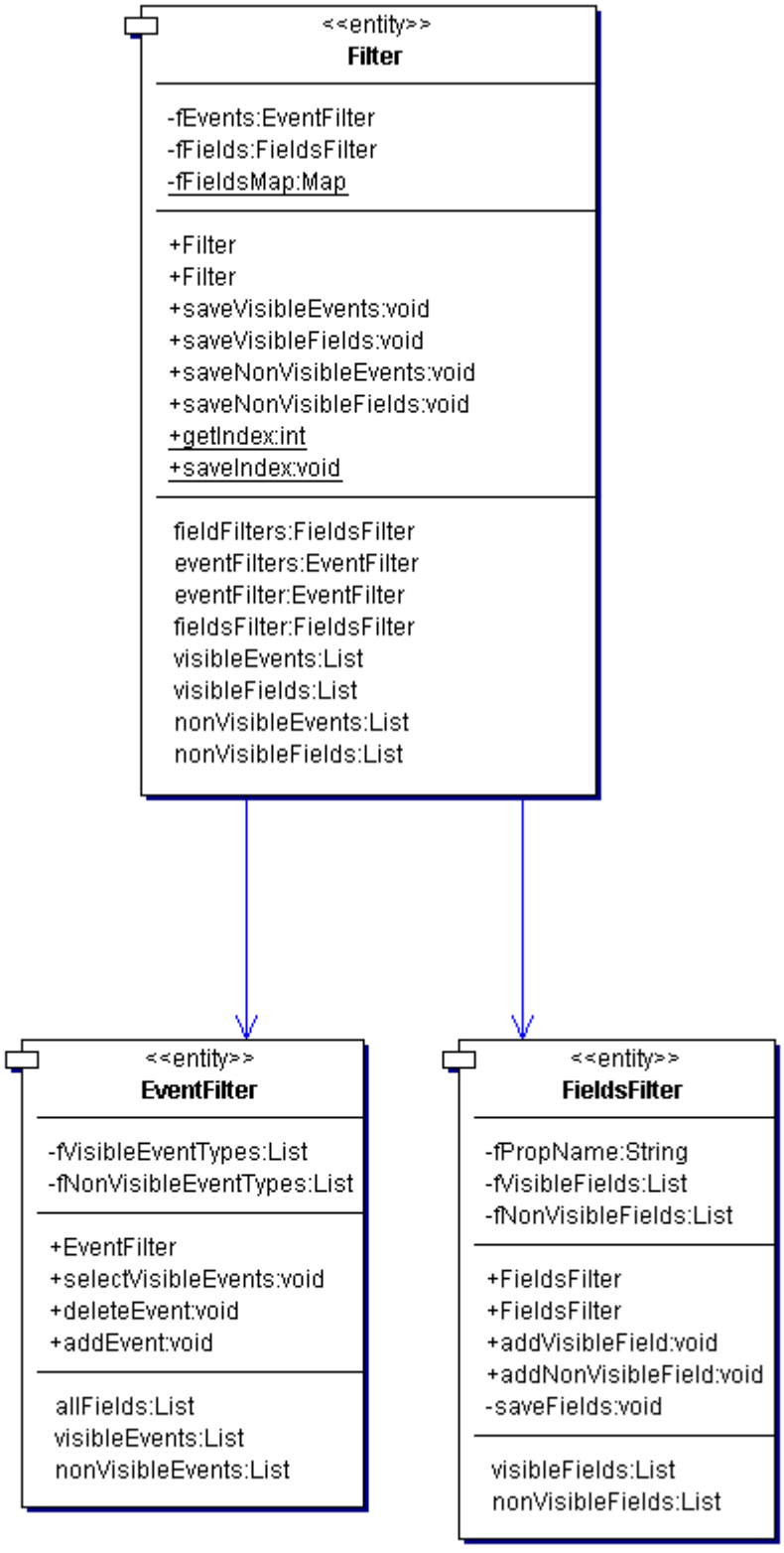


Figure 5-9: Print and Display Filters class diagram

5.2.1.10.1 Major Methods

EventFilter

- ***public void addEvent(EventType event)*** – adds the specified event to the list of filterable event types.

FieldsFilter

- ***public void setVisibleFields()*** – updates the visible field list to reflect the user selected visible fields.
- ***public List getVisibleFields()*** – retrieves the list of visible fields.

5.2.1.11 Report List

The Report List class maintains the names and locations of the input reports used during schedule generation. A separate String is maintained for each of the possible input reports. Each String contains both the report name and location (directory path).

Note that to generate a schedule, the user must enter a minimum of two reports, the TDRS Contact Report and the Orbital Events Report. An additional three reports: the SSR Buffer States Report; GN Report; and the ATC Load Report may also be necessary for schedule generation based on the options selected.

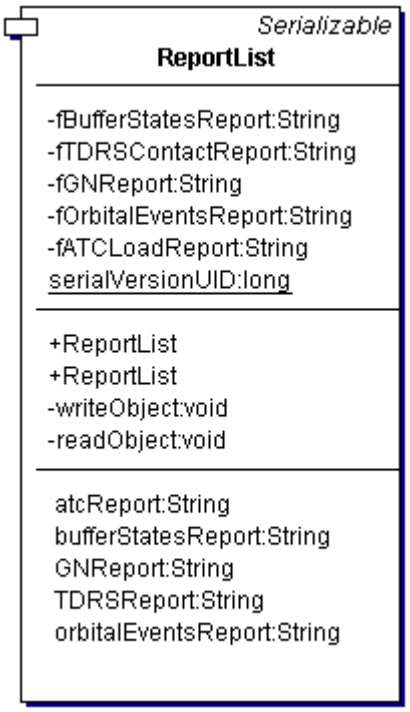


Figure 5-10: ReportList class diagram

5.2.1.11.1 Major Methods

- None

5.2.1.12 Report Manager

The Report Manager control class is responsible for managing all aspects of report ingestion and parsing. This includes retrieving input reports, parsing the input reports, setting the filterable events types, determining the initial dump windows, locating the synchronization point candidates and selecting an initial sync point.

Note that input files can be retrieved automatically from MMS or locally. The Report Manager stores the events resulting from file ingestion in the Event Contact Manager.

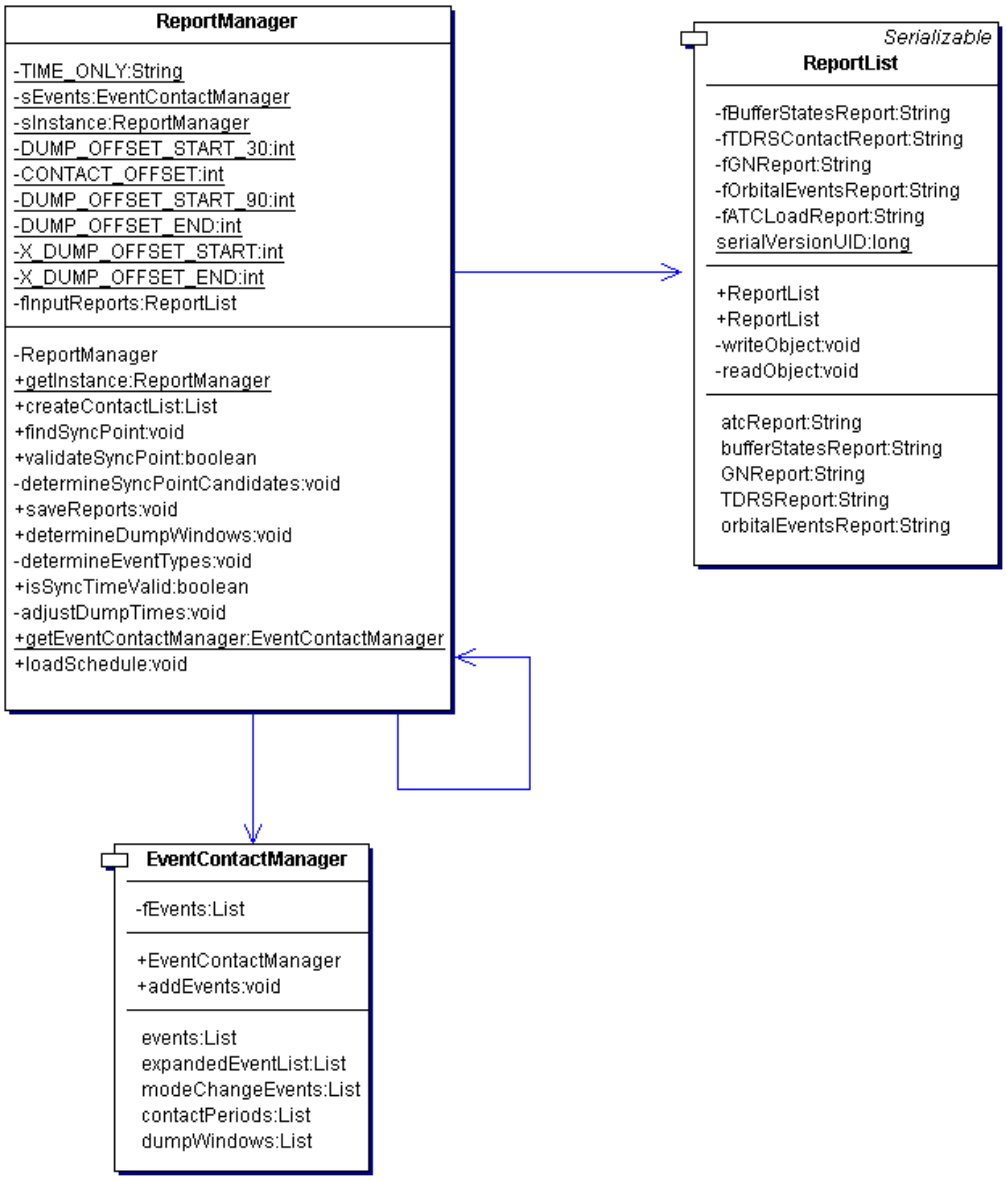


Figure 5-11: ReportManager class diagram

5.2.1.12.1 Major Methods

- ***public List createContactList()*** – coordinates and manages report parsing and construction of the event and contact list. The list of contacts, etc is returned after execution completes.
- ***public void findSyncPoint(List contactPeriods)*** – coordinates and controls selection of the synchronization point.
- ***public boolean validateSyncPoint(SyncPointParameters selectedSyncPoint)*** – validates that the selected sync point is in fact in a contact period long enough to completely empty the SSR buffers.
- ***public void determineSyncPointCandidates(String ssrBufferReport, Date start, Date end)*** – coordinates and controls the operation of determining the sync point candidates from the buffer states entries..
- ***public void determineDumpWindows(List contactPeriods, Date planStart)*** – coordinates and controls the automated Dump Window determination process.
- ***public void adjustDumpTimes(ContactWindow contact, int index, List contactPeriods)*** – adjust dump times as needed for overlapping contact periods.

5.2.1.13 Schedule Manager

The ScheduleManager control class is responsible for controlling and coordinating all operations that affect the Playback Schedule. The Schedule Manager provides support for generating the playback schedule, filtering the schedule for printing, printing the schedule, and saving the schedule to a file.

Figure 5-12: ScheduleManager class diagram

5.2.1.13.1 Major Methods

- ***public List generateSchedule()*** – creates the playback schedule for display and/or printing.
- ***public List schedulePlaybacks()*** – creates playback windows for each of the dump windows in the scheduling horizon.
- ***public PlayBackSchedule getSchedule()*** – retrieves the full unfiltered playback schedule.

5.2.1.14 Scheduling Options

The Scheduling Options class contains user provided information regarding the scheduling window, report locations, etc. The information contained in this class includes: the scheduling window start and stop; report locations, the ASTER modeling mode, etc. These options are used during report parsing and schedule generation.

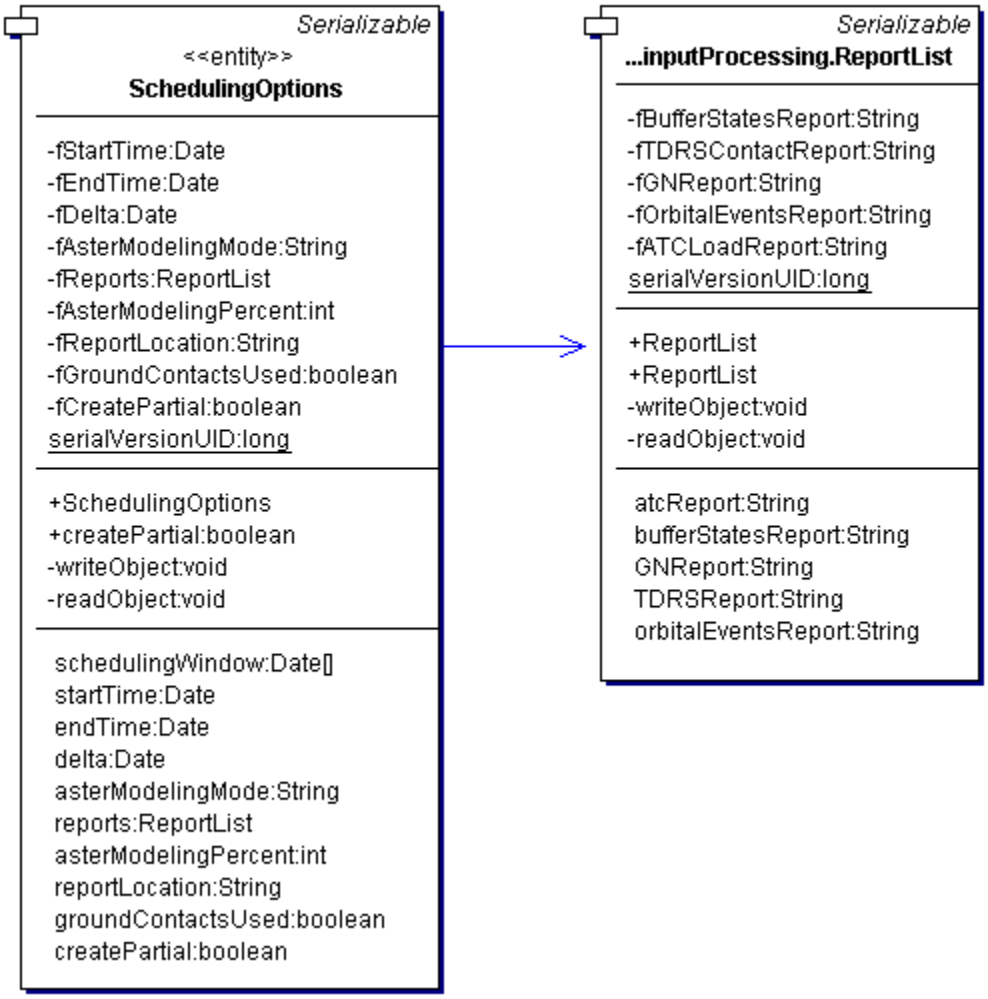


Figure 5-13: SchedulingOptions class diagram

5.2.1.14.1 Major Methods

- ***public Date[] getSchedulingWindow()*** – returns the scheduling window start time, end time, and delta in a Date array.

5.2.1.15 Sync Point Parameters

The Sync Point Parameters class is responsible for maintaining the list of synchronization point candidates for the delta window after the start of the scheduling period. Each Sync Point entry contains the information for a candidate synchronization point and includes a time stamp, the buffer capacities, and the buffer durations for the ASTER, MISR, and MODIS buffers. From these candidate entries the ASSET tool selects a single Sync Point Parameter entry as the synchronization point. The synchronization point represents the earliest contact after the start of the scheduling window up to the delta time in which the SSR buffers can be completely emptied.

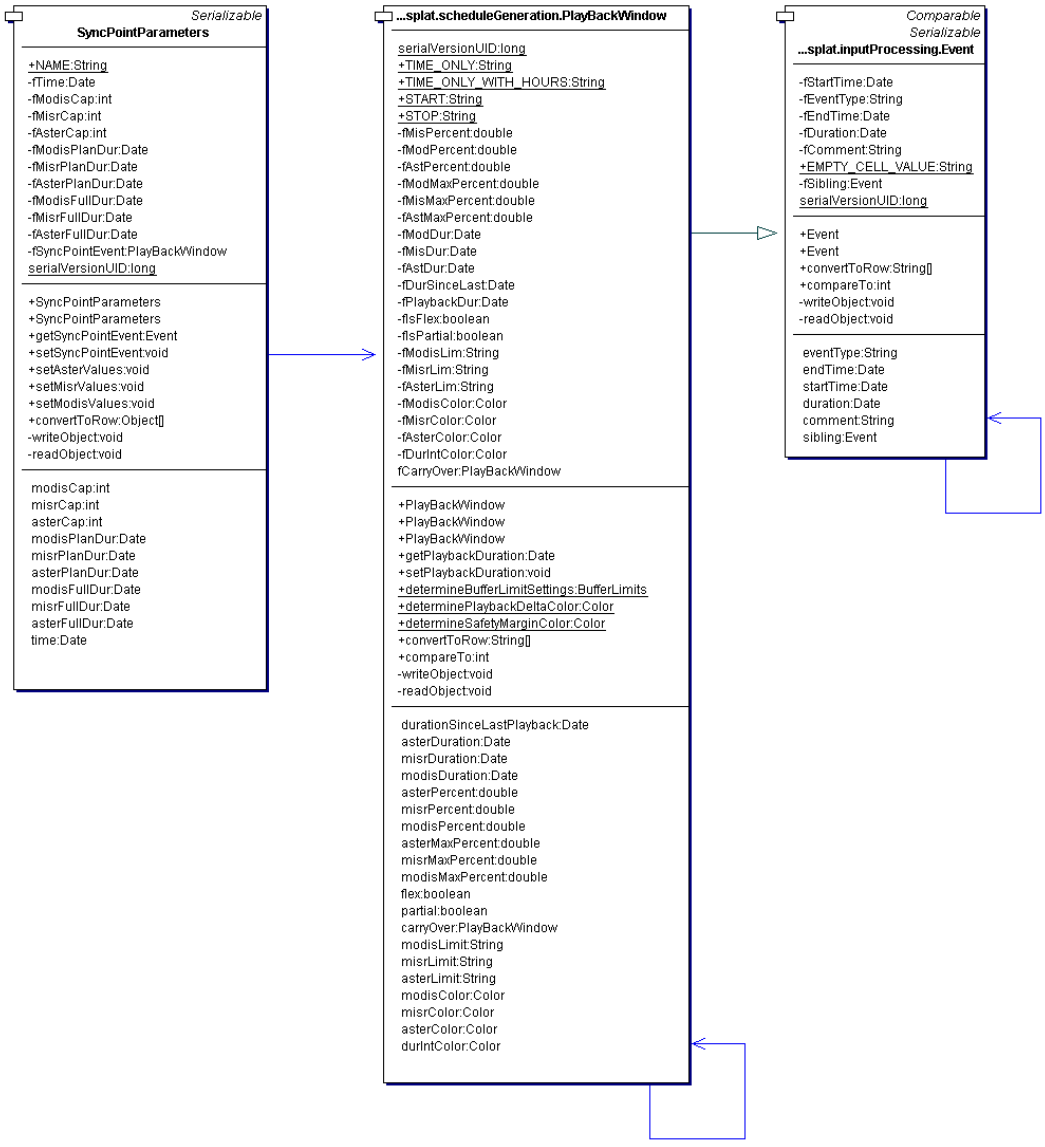


Figure 5-14: SyncPointParamaters class diagram

5.2.1.15.1 Major Methods

- ***public Event getSyncPointEvent()*** – returns the timeline event for this sync point.
- ***public void setSyncPointEvent(PlayBackWindow syncPoint)*** – sets the event (playback) for this sync point.
- ***public String[] convertToRow()*** – returns the content of the associated SyncPoint event as an array of Strings for display or output.

5.2.1.16 Print Manager

The PrintManager class is responsible for coordinating and managing all aspects of printing and previewing playback schedules. The PrintManager provides functionality for previewing and printing to a file or printer.

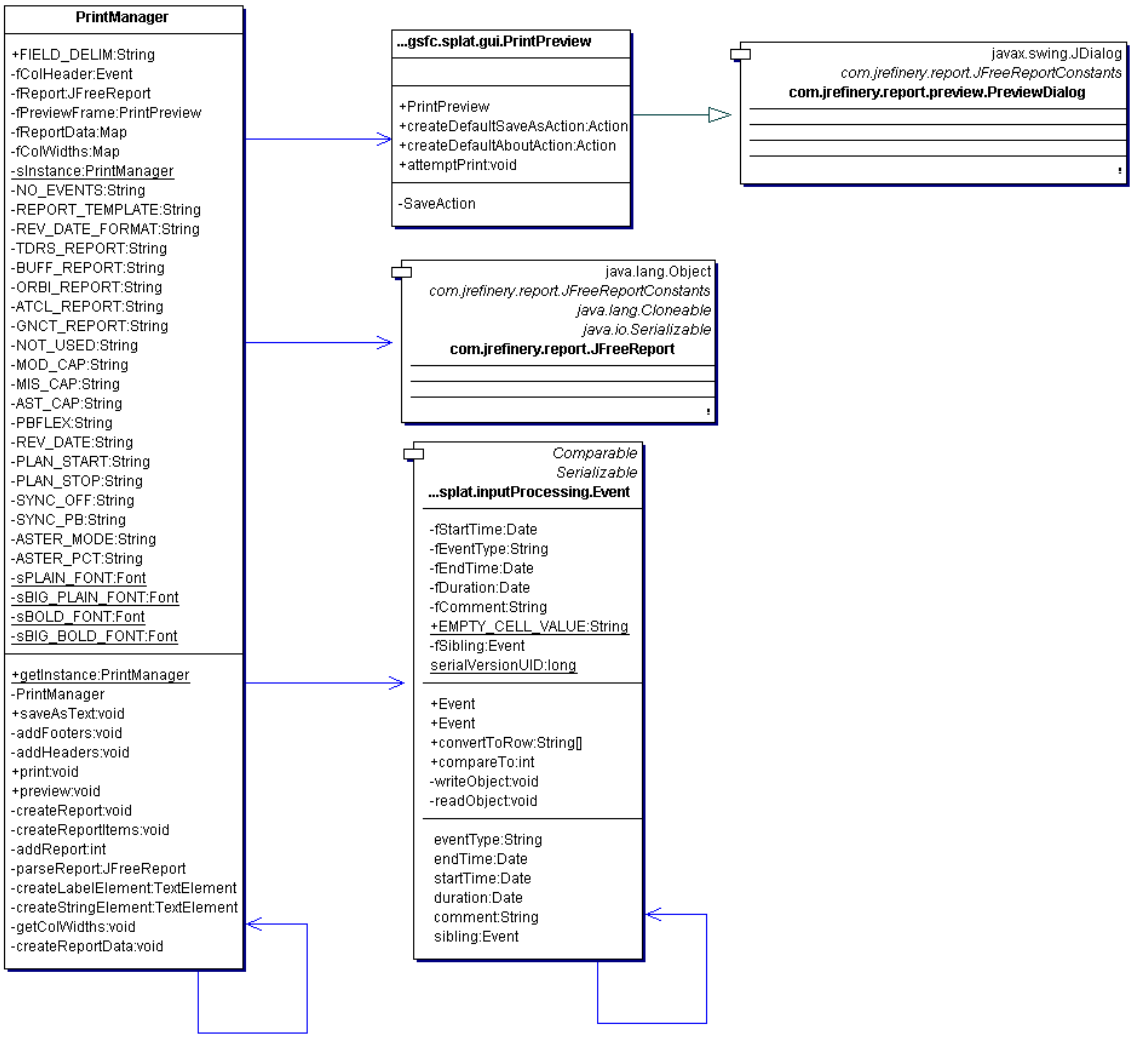


Figure 5-15: PrintManager class diagram

5.2.1.16.1 Major Methods

- ***public void print(String addText)*** – prints the current PlaybackSchedule.
- ***public void preview(String addText)*** – provides a print preview of the current PlayBackSchedule.
- ***public void saveAsText(String filename)*** – formats and saves the current schedule in the specified text file.

5.2.1.17 Recent File Manager

The RecentFileManager class is responsible for managing and maintaining the list of recent schedules opened in the ASSET tool.

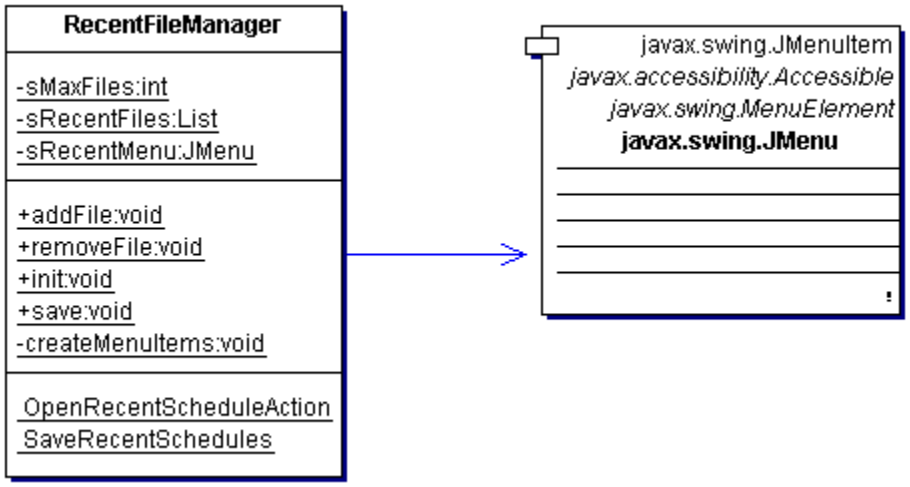


Figure 5-16: RecentFileManager class diagram

5.2.1.17.1 Major Methods

- ***public static void addFile(File file)*** – add the specified filename to the list of recent files.
- ***public static void removeFile(File file)*** – remove the specified file from the list of recent files.
- ***public void save()*** – save the list of recent files to the system properties.

5.2.1.18 ASTER Mode Entry

The ASTER Mode entry class is responsible for maintaining a single ASTER RTCS macro definition. Each instance of this class contains information regarding which instruments in the ASTER will turn on and off and at what times.

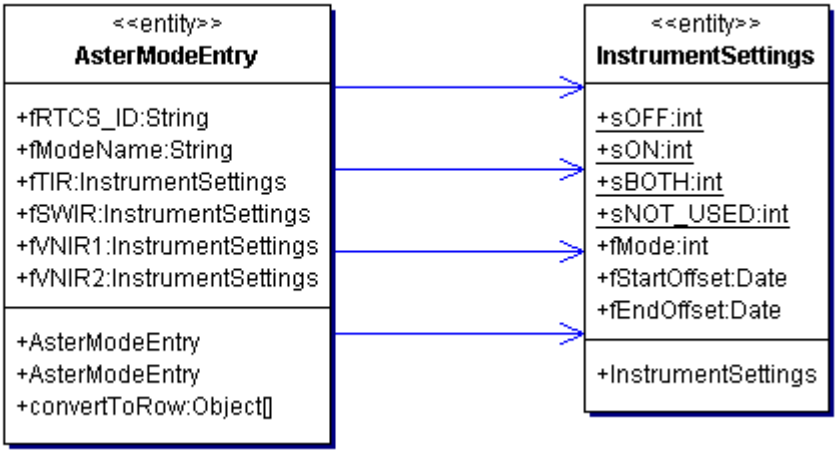


Figure 5-17: AsterModeEntry class diagram

5.2.1.18.1 Major Methods

public Object[] convertToRow() – converts the content of this object to an array of Objects for display in a dialog.

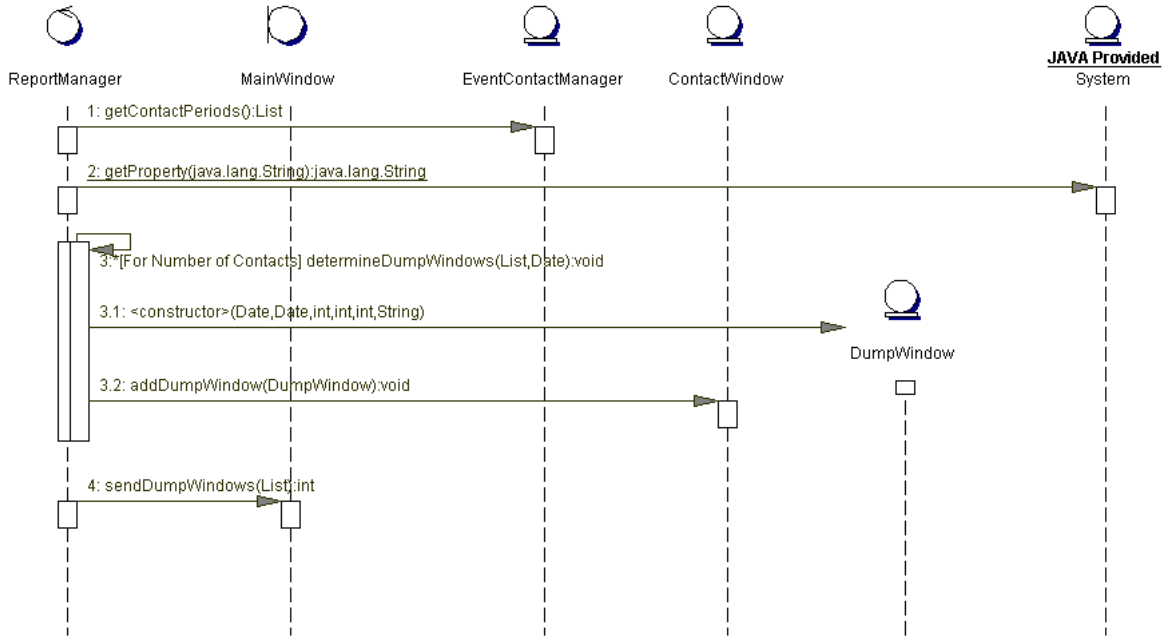
5.2.2 Sequence Diagrams

This section contains sequence diagrams that describe the interactions among classes and components within the ASSET Tool. The sequence diagrams are separated by system function. Sequence diagrams are provided for report processing, schedule generation, automated dump window determination, and automated sync point determination.

Note that persistent data storage for the ASSET system is being provided by the Java Application Shell (JAS). Sequence diagrams are not provided for JAS features.

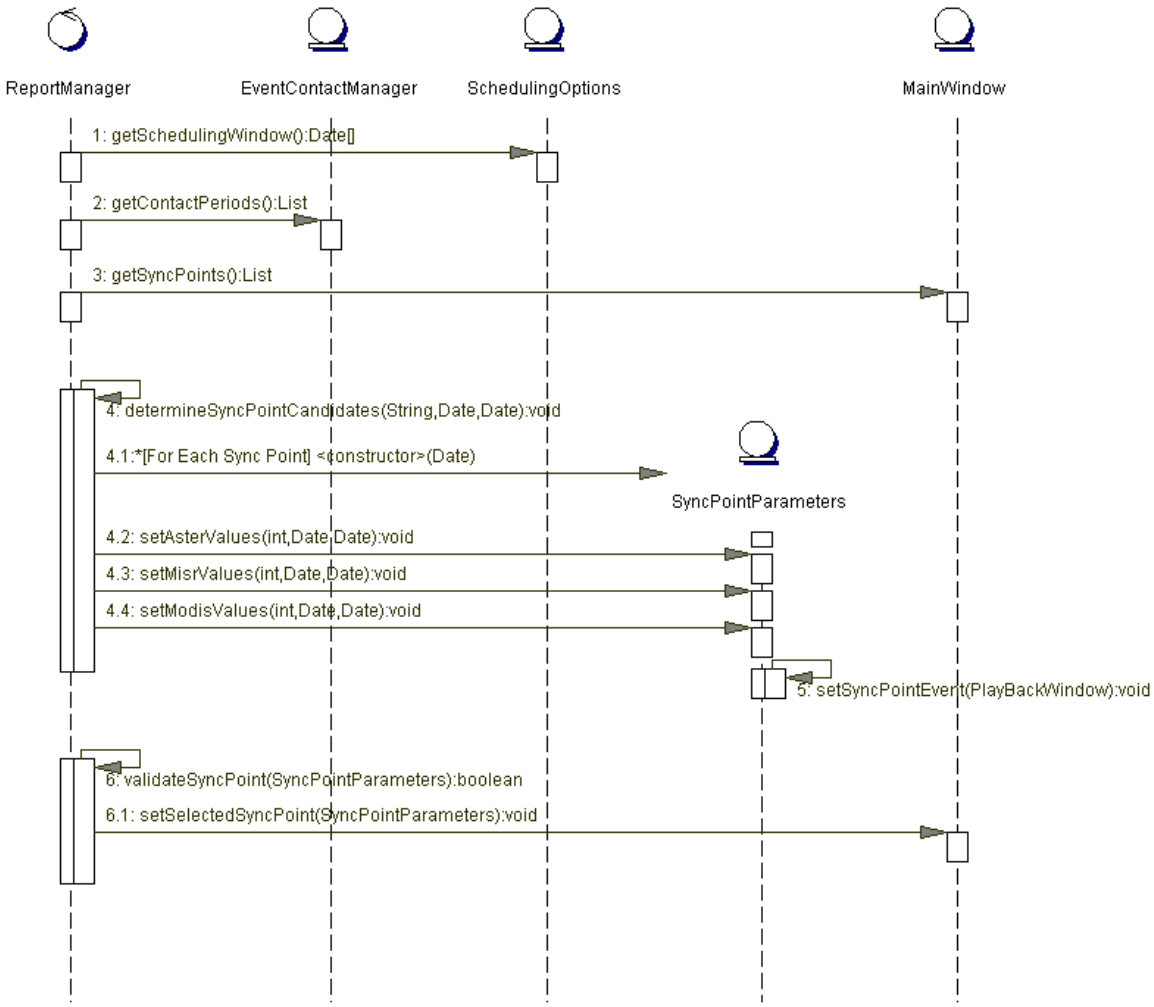
5.2.2.1 Determine Dump Windows Sequence Diagram

The following sequence diagram describes the interactions among classes during dump window determination. The automated dump window determination process involves the ASSET tool automatically selecting the initial set of dump windows based on the contact periods extracted from the Input Reports, a list of schedulable contact types, and user provided dump window offsets.



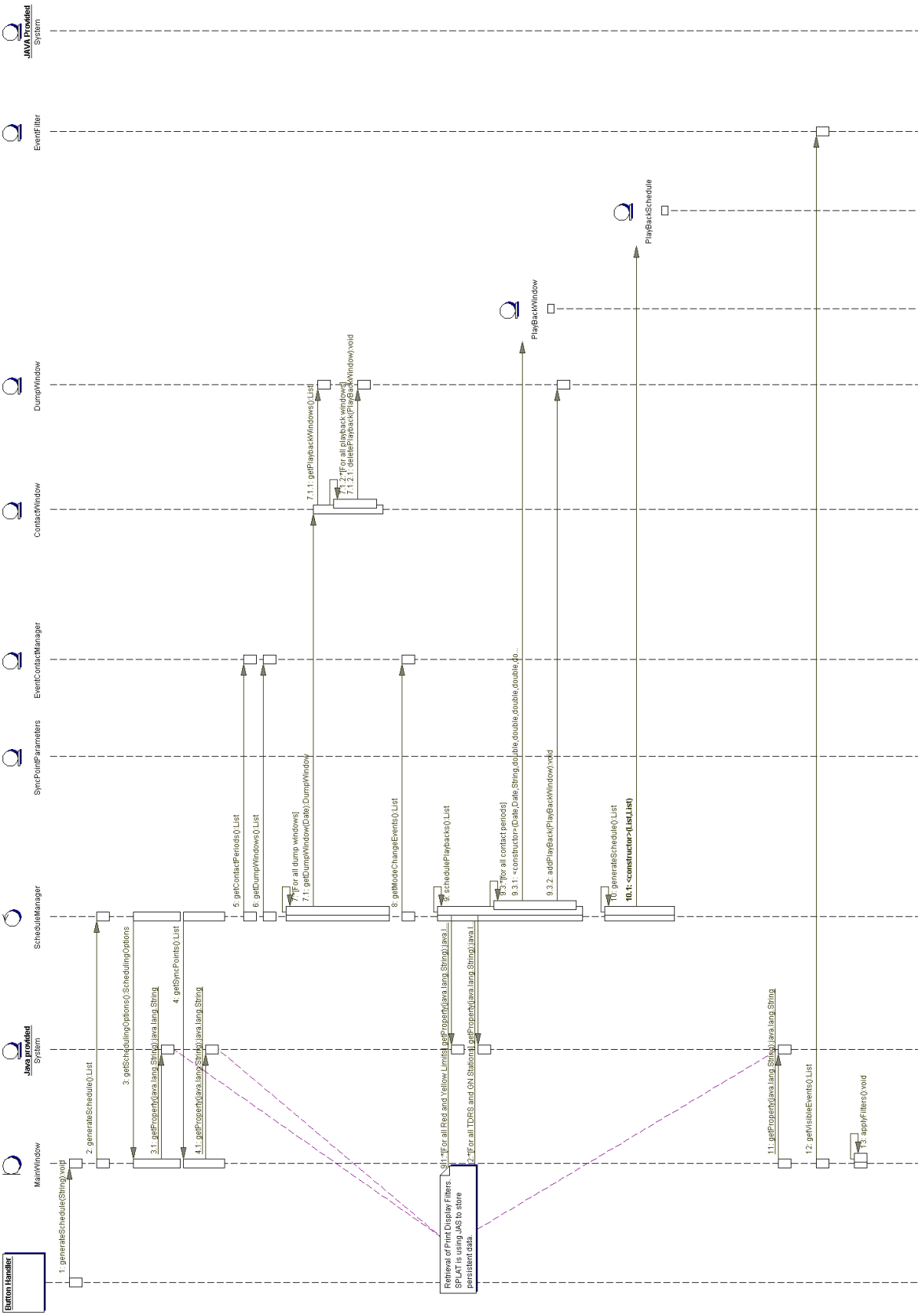
5.2.2.2 Determine Sync Point Sequence Diagram

The following sequence diagram describes the interactions among classes during sync point determination. Sync Point determination involves the ASSET tool automatically selecting the sync point candidates within the delta window and selecting a single initial synchronization point from the candidate list.



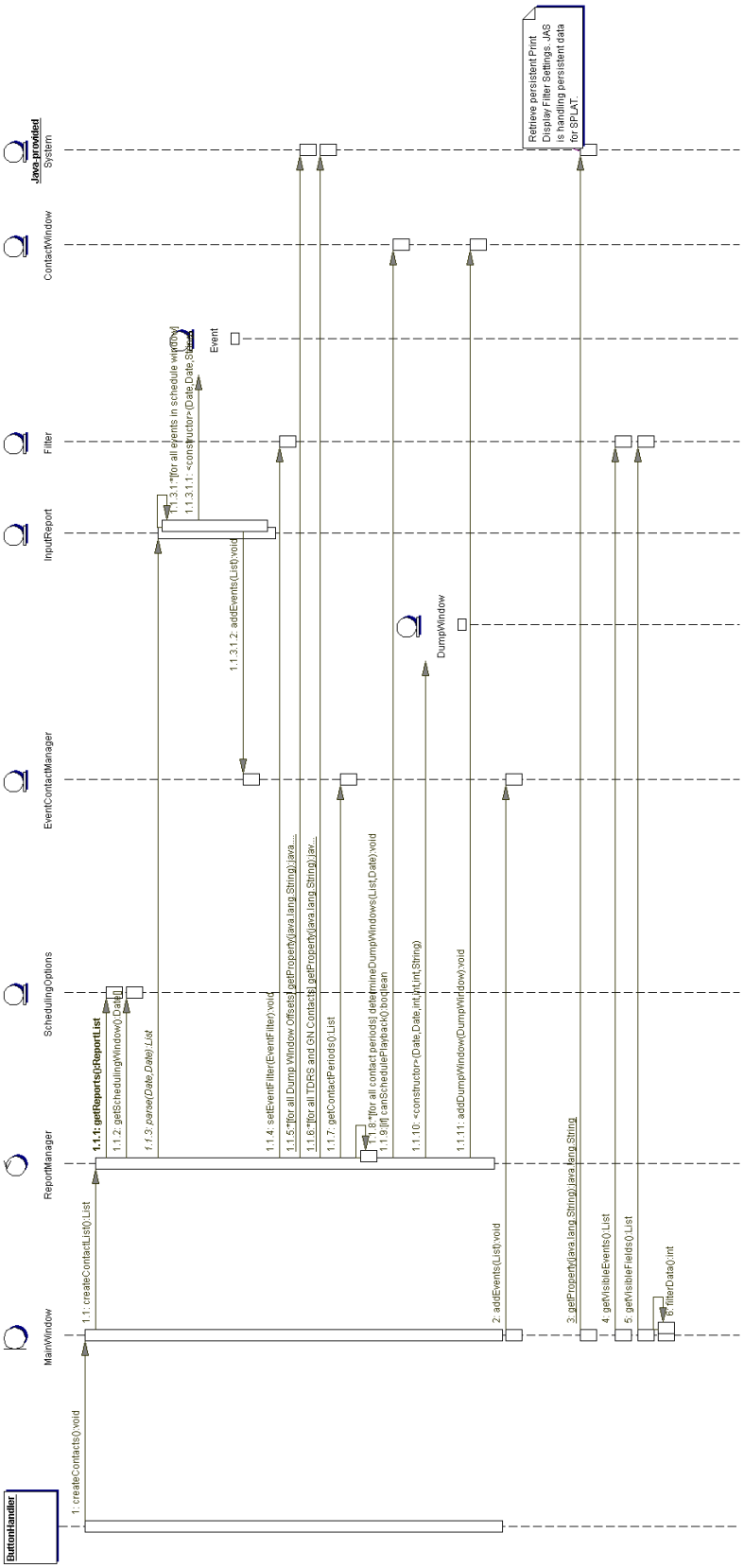
5.2.2.3 Generate SSR Buffer Playback Schedule

The following rather large sequence diagram describes the interactions among classes during playback schedule generation. Playback Schedule generation involves determining the playback window locations and durations for each of the dump windows based on buffer usage and mode changes.


Button Handler
MainWindow
Java provided System
ScheduleManager
SyncPointParameters
EventContactManager
ContactWindow
DumpWindow
EventFilter
JAVA Provided System
PlayBackWindow
PlayBackSchedule
1: generateSchedule(String):void
2: generateSchedule():List
3: getSchedulingOptions():SchedulingOptions
3.1: getProperty(java.lang.String):java.lang.String
4: getSyncPoints():List
4.1: getProperty(java.lang.String):java.lang.String
5: getContactPeriods():List
6: getDumpWindows():List
7: [For all dump windows]
7.1: getDumpWindow(Date):DumpWindow
7.1.1: getPlaybackWindows():List
7.1.2: [For all playback windows]
7.1.2.1: deletePlayback(PlayBackWindow):void
8: getModeChangeEvents():List
9: scheduePlaybacks():List
9.1: [For all Red and Yellow Limits] getProperty(java.lang.String):java.l...
9.2: [For all TDRS and GN Stations] getProperty(java.lang.String):java.l...
9.3: [For all contact periods]
9.3.1: <constructor>(Date,Date,String,double,double,double,double,do...
9.3.2: addPlayBack(PlayBackWindow):void
10: generateSchedule():List
10.1: <constructor>(List,List)
11: getProperty(java.lang.String):java.lang.String
12: getVisibleEvents():List
13: applyFilters():void
Retrieval of Print Display Filters SPLAT is using JAS to store persistent data.

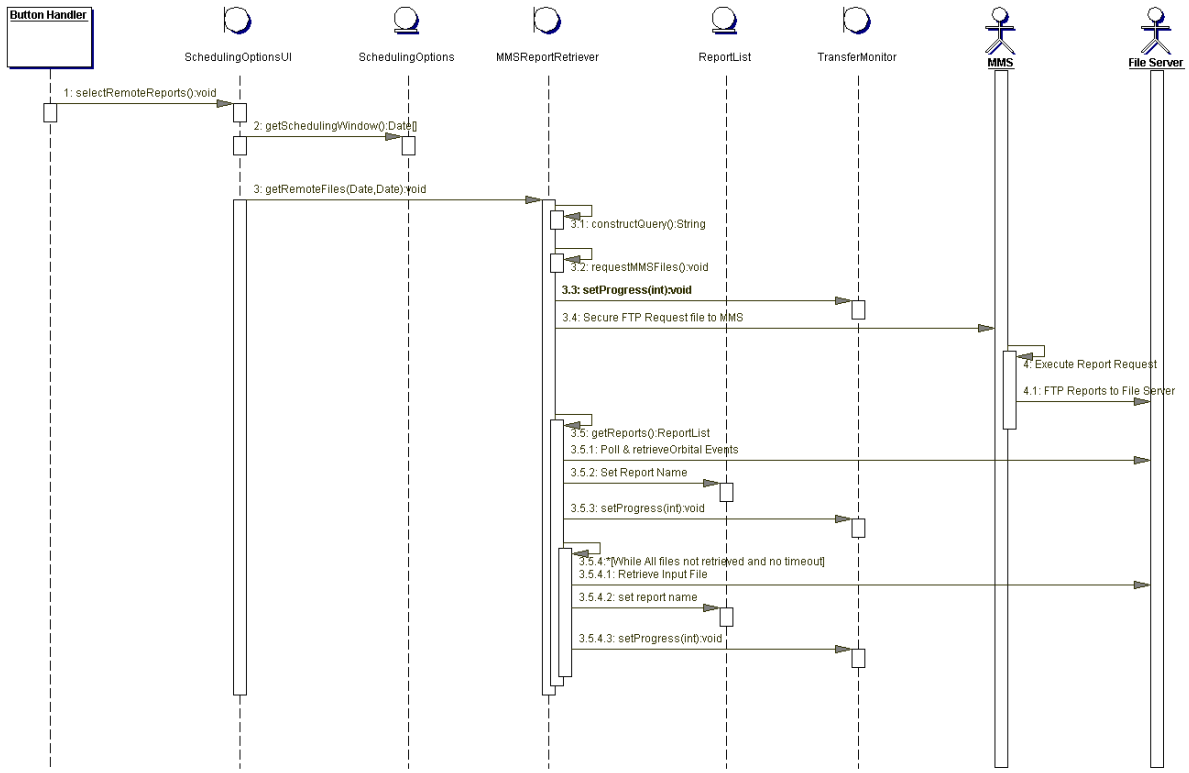
5.2.2.4 Process Input Reports Sequence Diagram

The following sequence diagram describes the interactions among classes during report processing. Report Processing involves retrieving and parse the individual input reports, extracting mode changes, contact periods, orbital events, ASTER imaging events.


ButtonHandler
MainWindow
ReportManager
SchedulingOptions
EventContactManager
InputReport
Filter
Event
ContactWindow
Java pr ovided System
DumpWindow
Retrieve persistent Print Display Filter Settings. JAS is handling persistent data for SPLAT.
1: createContacts():void
1.1: createContactList():List
1.1.1: getReports():ReportList
1.1.2: getSchedulingWindow():Date
1.1.3: parse(Date,Date):List
1.1.4: setEventFilter(EventFilter):void
1.1.5: *[for all Dump Window Offsets] getProperty(java.lang.String):java....
1.1.6: *[for all TDRS and GN Contacts] getProperty(java.lang.String):jav...
1.1.7: getContactPeriods():List
1.1.8: *[for all contact periods] determineDumpWindows(List,Date):void
1.1.9: [if] canScheduledPlayback():boolean
1.1.10: <constructor>(Date,Date,int,int,int,String)
1.1.11: addDumpWindow(DumpWindow):void
1.1.3.1: *[for all events in schedule window]
1.1.3.1.1: <constructor>(Date,Date,Status)
1.1.3.1.2: addEvents(List):void
2: addEvents(List):void
3: getProperty(java.lang.String):java.lang.String
4: getVisibleEvents():List
5: getVisibleFields():List
6: filterData():int

5.2.2.5 Report Retrieval Sequence Diagram – Automated

The following sequence diagram describes the interactions among classes during automated report retrieval. Automated Report retrieval involves the creation and delivery to MMS of a report request file, the MMS generation and delivery of the report files to a file server and the subsequent retrieval of the report files by ASSET from the file server.



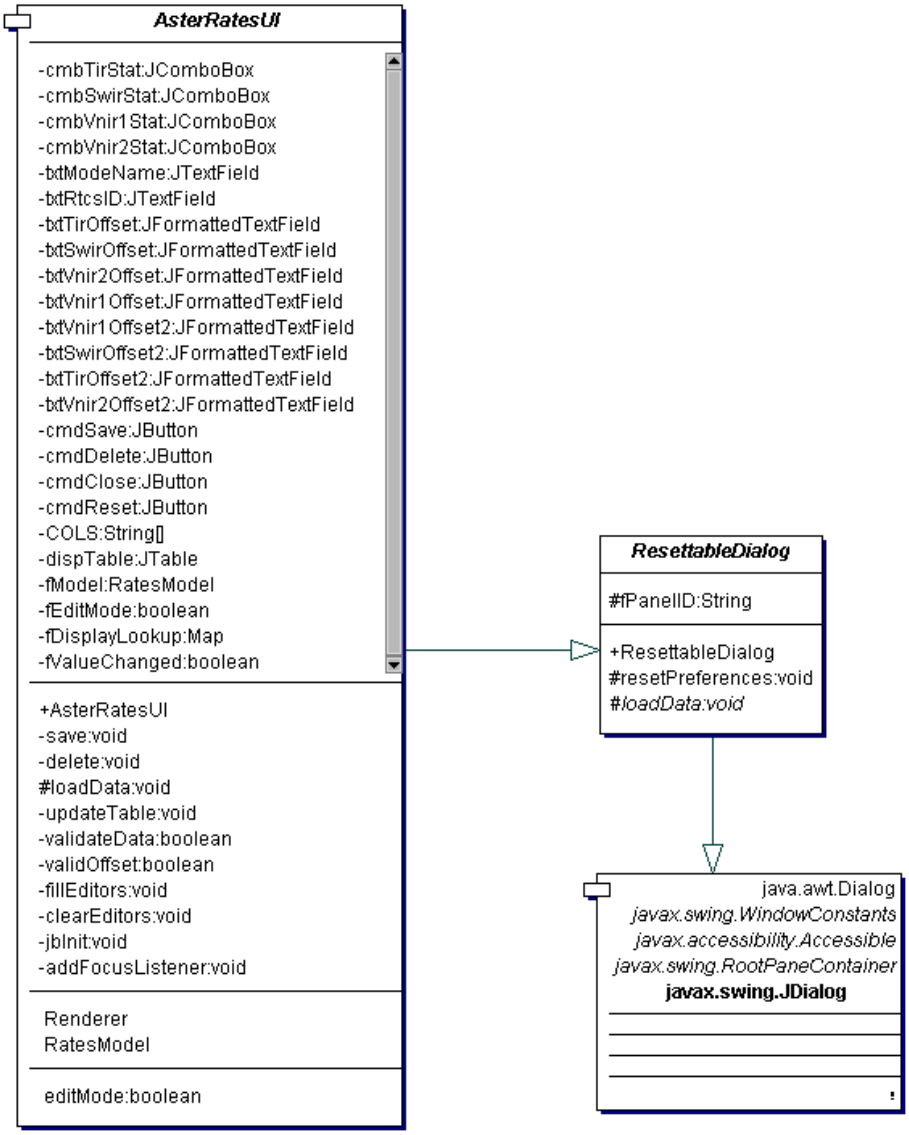
6 User Interface Design

This section includes descriptions and class diagrams for each of the GUI components in the ASSET system. For each GUI, a class diagram depicting the GUI components and one or more sequence diagrams depicting interaction with other classes in the system is provided.

6.1 ASTER Rates UI

The ASTER Rates UI is a dialog through which the user can modify existing ASTER imaging rates or add new imaging modes. Through this interface the user can select the name of the modeling mode, the RTCS_ID for the mode, which instruments are active, and when the instruments turn on or off.

6.1.1 ASTER Rates UI Class Diagram

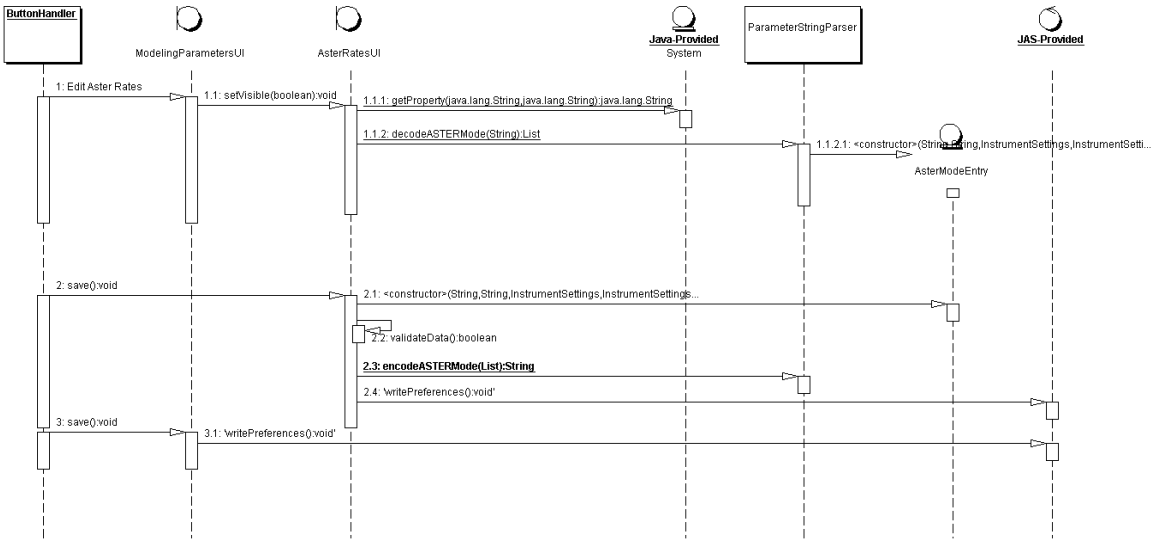


6.1.1.1 Major Methods

- ***public void save()*** – saves the modified or new ASTER modes of operations.
- ***public void delete()*** – removes the current rate from the table..
- ***public boolean validateData()*** – validates the user provided rate data.

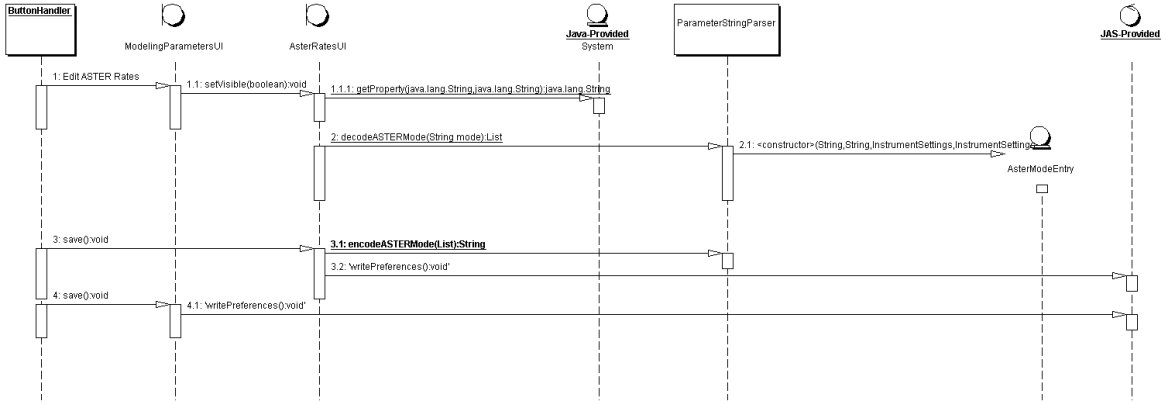
6.1.2 Add an ASTER Mode Sequence Diagram

The following sequence diagram describes the interactions among classes when creating a new ASTER imaging mode.



6.1.3 Edit an ASTER Imaging Mode

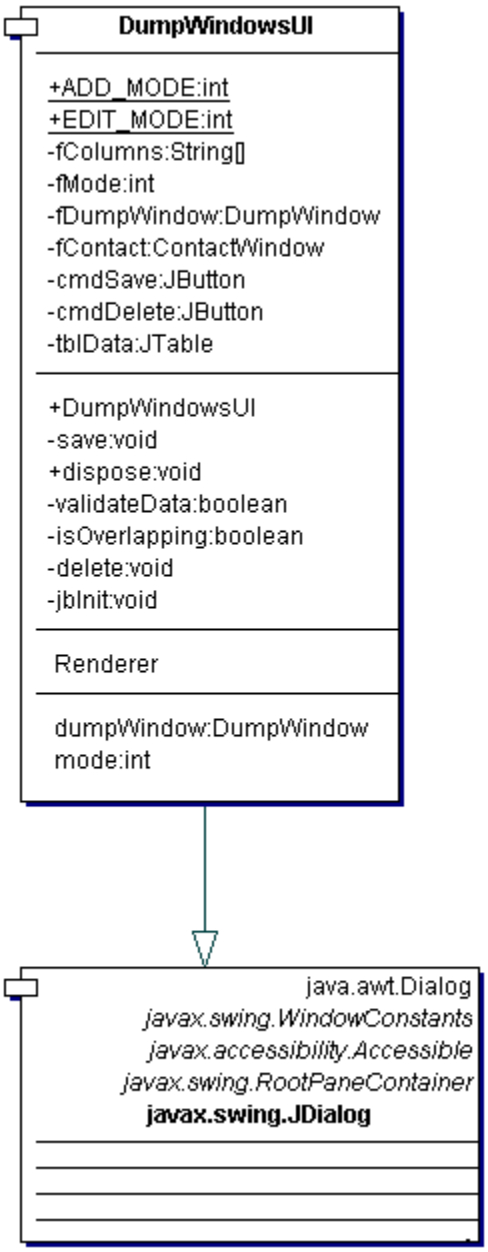
The following sequence diagram describes the interactions among class when editing an ASTER imaging mode.



6.2 Dump Windows UI

The Dump Windows UI is a JDialog displayed when the user wants to modify or add dump windows. Through this dialog box the user performs one of the following options: Add a new Dump Window; edit an existing Dump Window; or delete an existing Dump Window.

6.2.1 Dump Windows UI Class Diagram

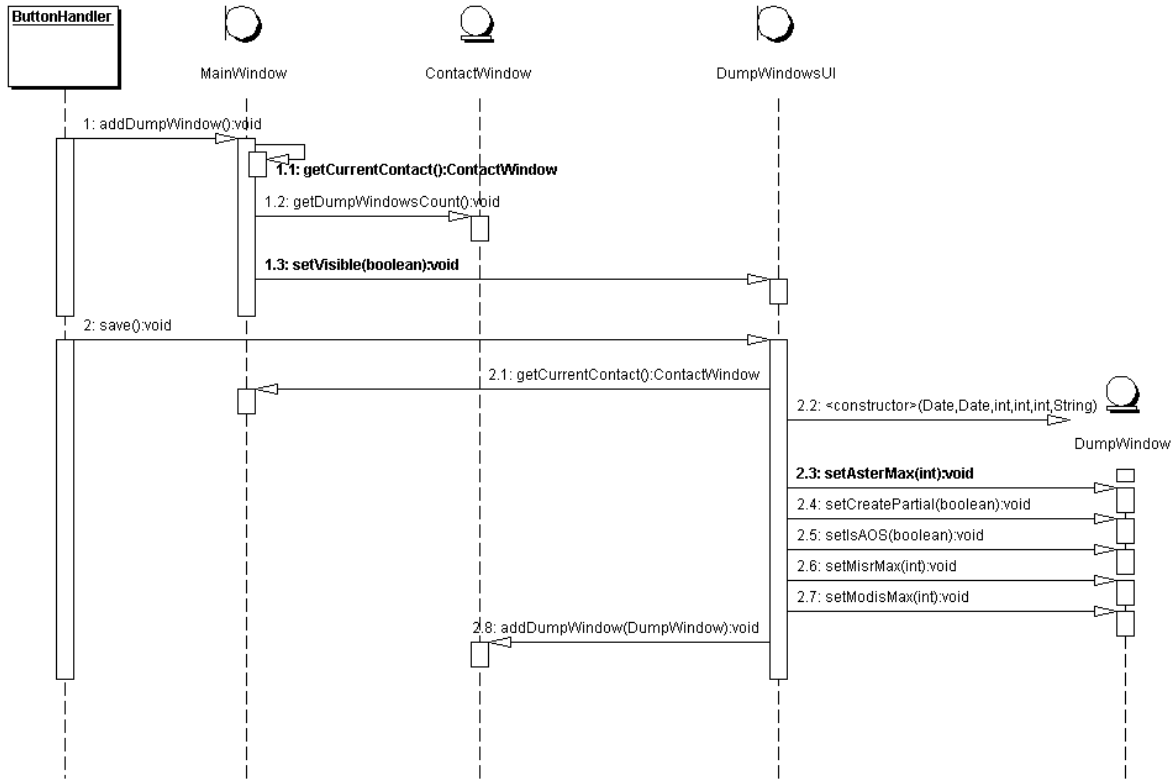


6.2.1.1 Major Methods

- ***public void save()*** – stores the modified values to the Dump Window.
- ***public void dispose()*** – removes the currently displayed Dump Window entry.
- ***public Boolean validate()*** – validates the user provided data.
- ***public Boolean isOverlapping(Date start, Date end)*** – check for overlap with other dump windows in the contact.

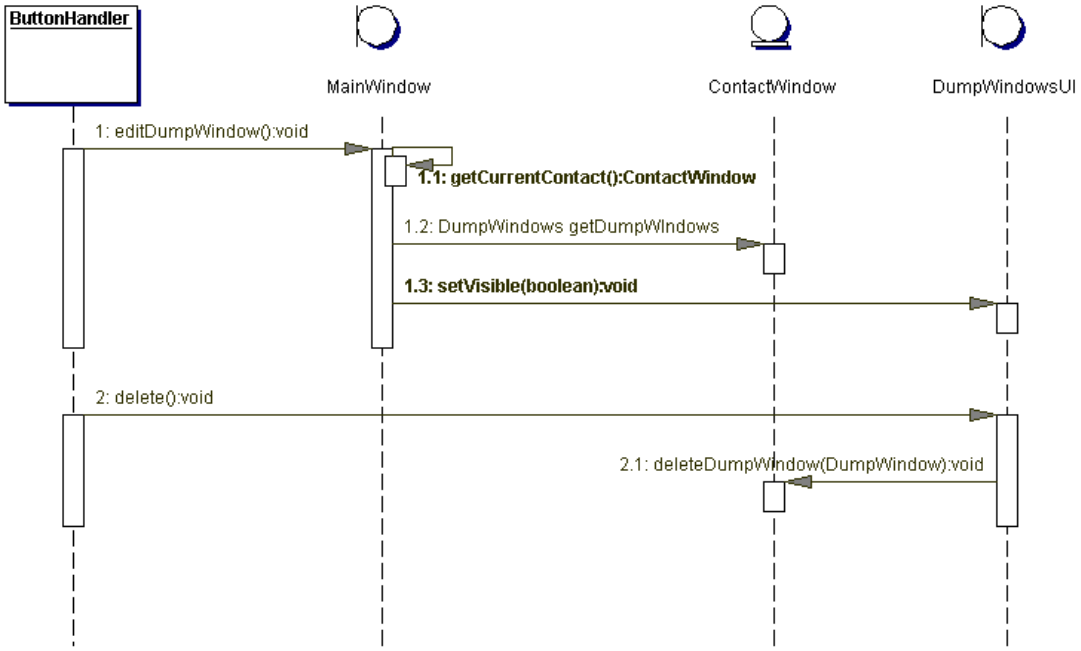
6.2.1.2 Add a Dump Window Sequence Diagram

The following sequence diagram describes the interaction among classes when adding a new dump window to a contact.



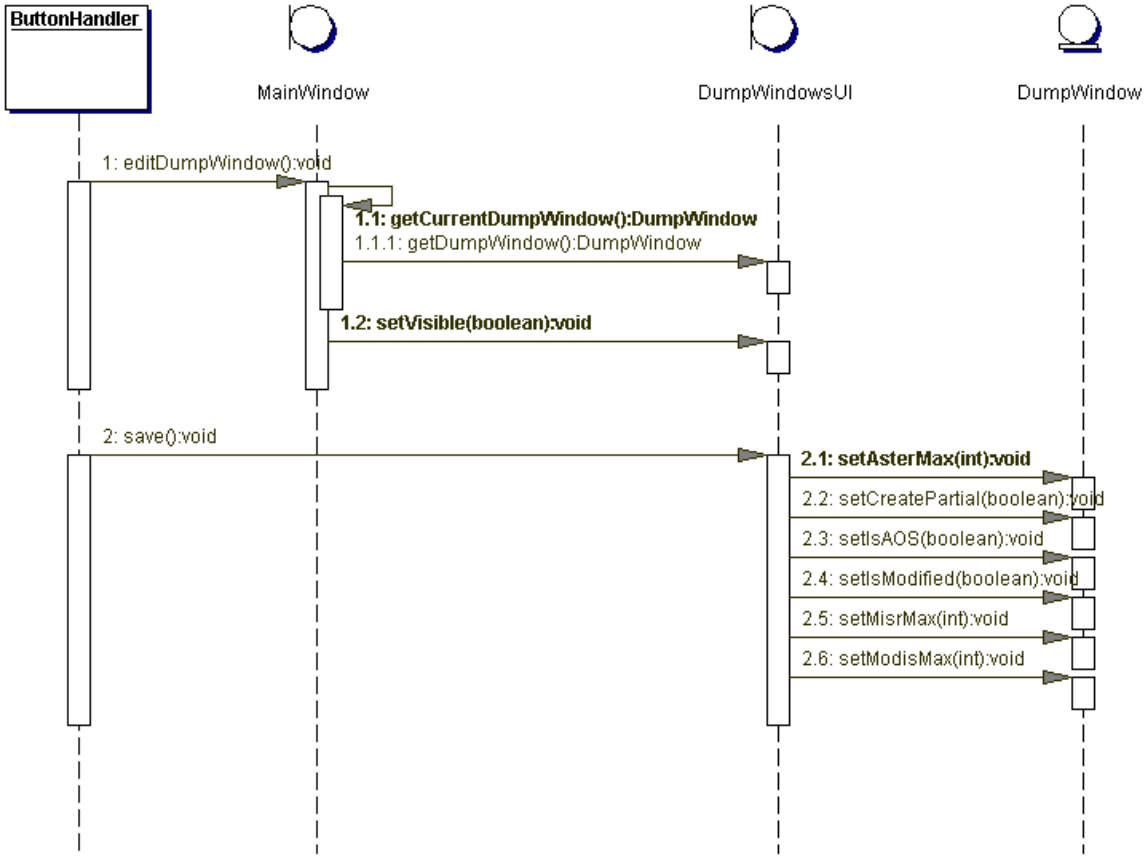
6.2.2 Delete a Dump Window Sequence Diagram

The following sequence diagram describes the interactions among classes when removing an existing dump window from a contact.



6.2.3 Edit a Dump Window Sequence Diagram

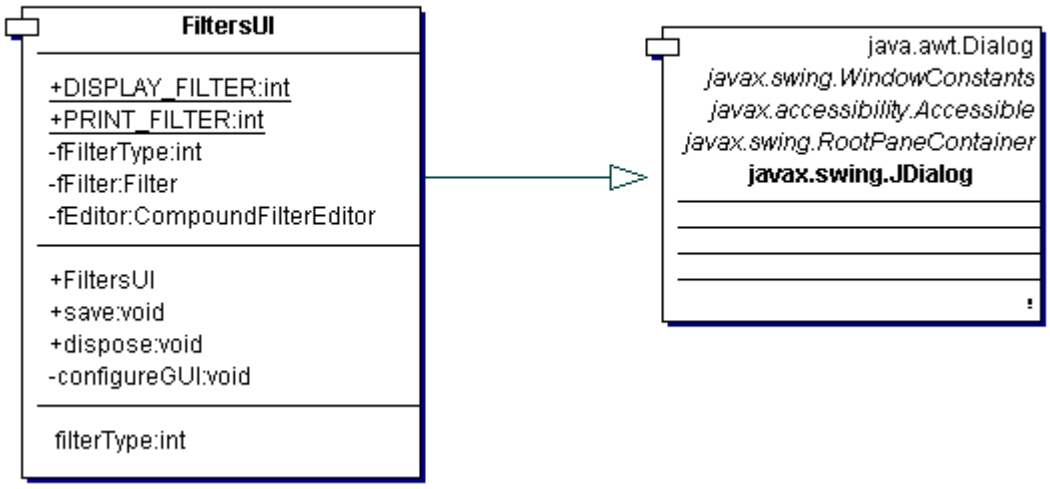
The following sequence diagram describes the interactions among classes when editing a dump window entry.



6.3 Filter UI

The Filter UI is a configurable User Interface that provides a common front end to both print and display filter options. Through this GUI, the user selects which fields and event types to display and/or print depending on which filter type is being edited.

6.3.1.1 Filter UI Class Diagram

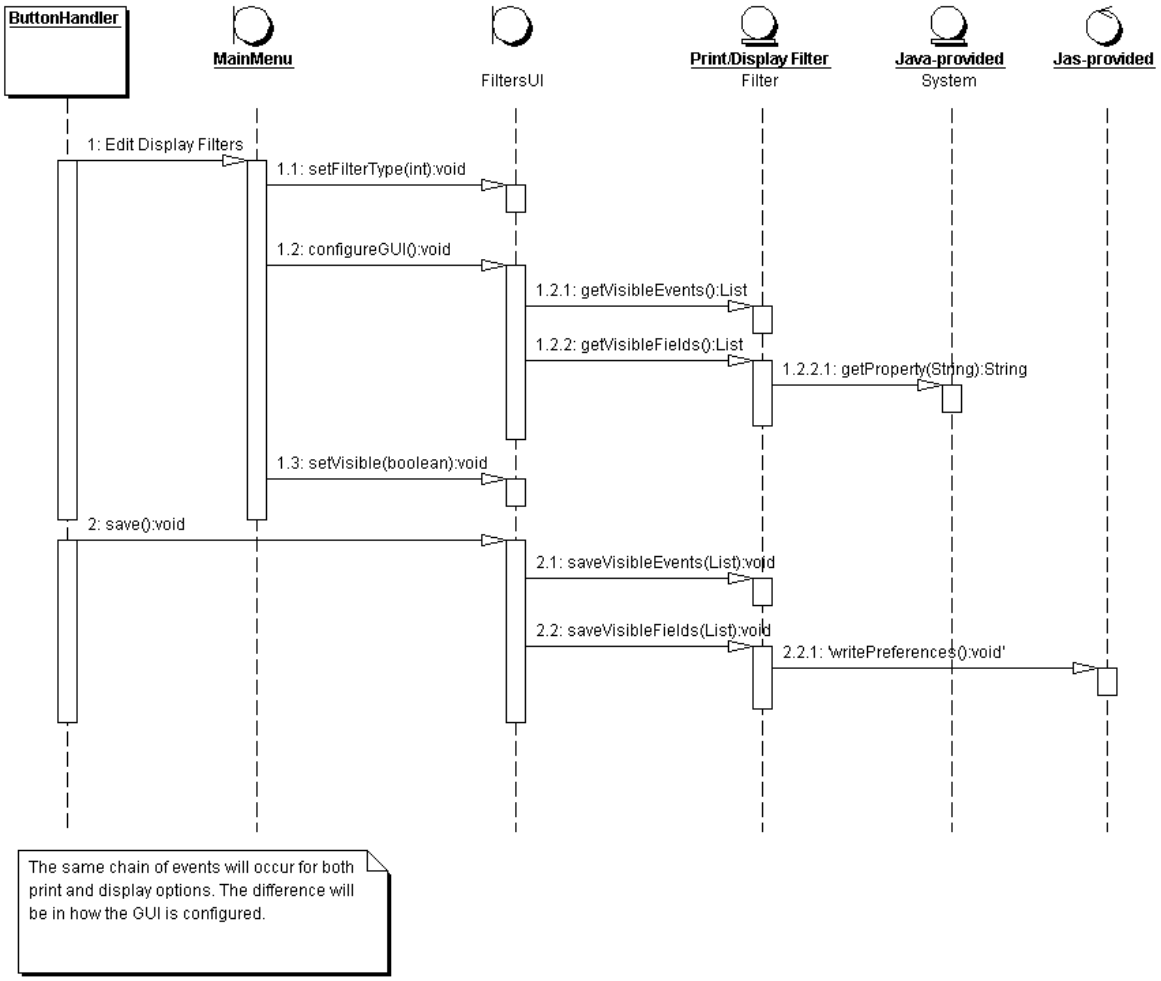


6.3.1.2 Major Methods

- ***public void save()*** – saves the modified filter settings.
- ***public void configureGUI()*** – configures the GUI contents based on the type of Filter (print or display).

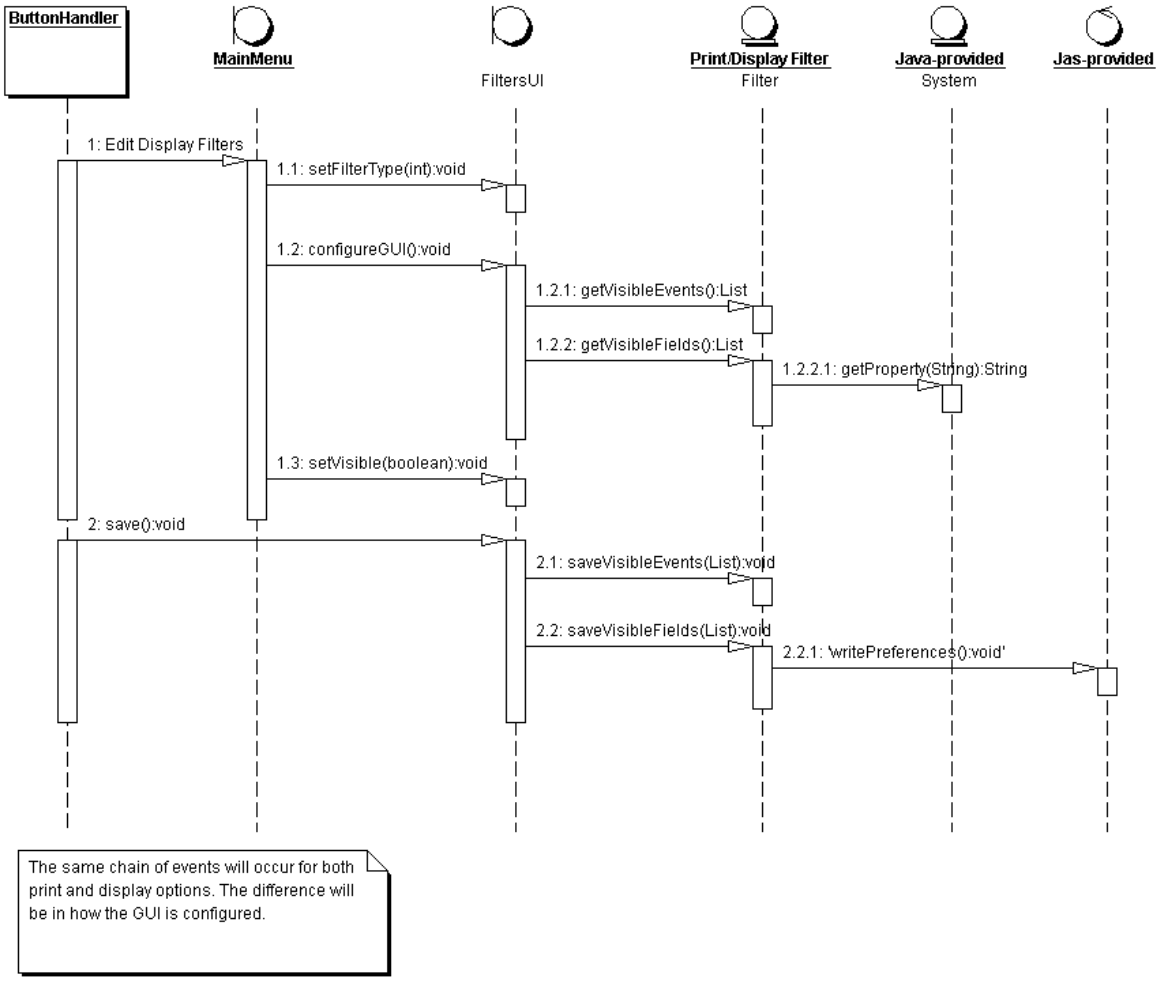
6.3.2 Edit Display Filters Sequence Diagram

The following sequence diagram describes the interactions among classes when editing the display filters.



6.3.3 Edit Print Filters Sequence Diagram

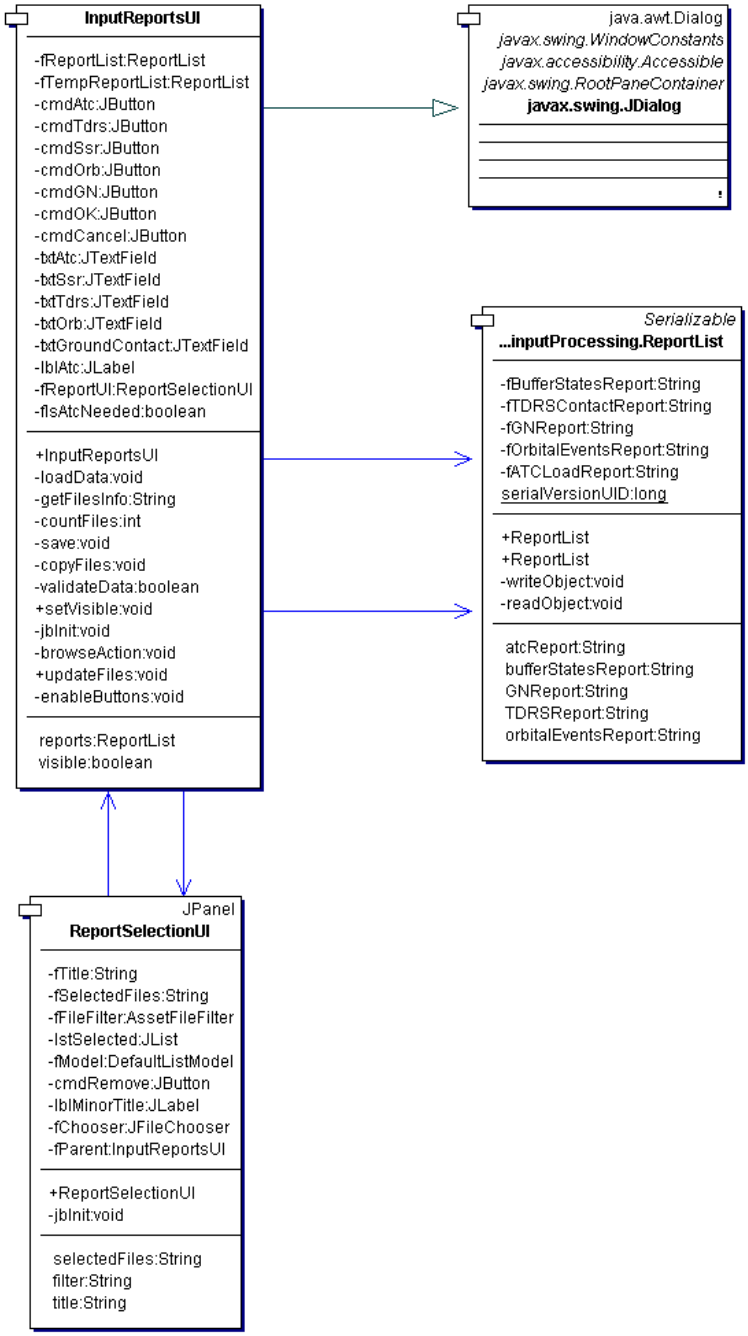
The following sequence diagram describes the interactions among classes when editing the print filters.



6.4 Input Reports UI

The Input Reports UI provides an interface through which the user selects the names and locations of the input reports necessary for schedule generation on the local machine.

6.4.1.1 Input Reports UI Class Diagram

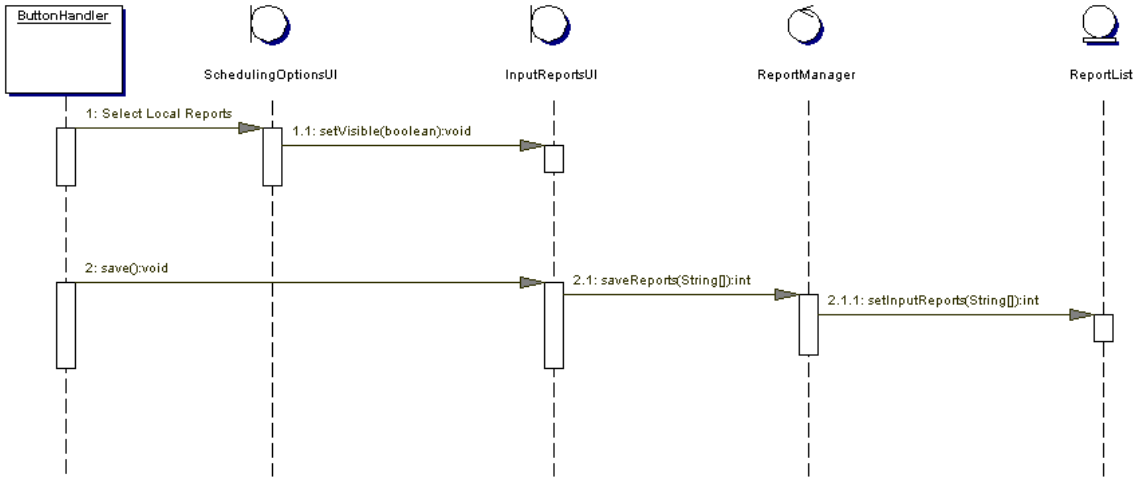


6.4.1.2 Major Methods

- ***public void loadData()*** – sets the current values for the input reports.
- ***public void save()*** – saves the user provided/modified report names and locations.
- ***public boolean validateDate()*** – validates the user provided report information.

6.4.2 Input Reports UI Sequence Diagram

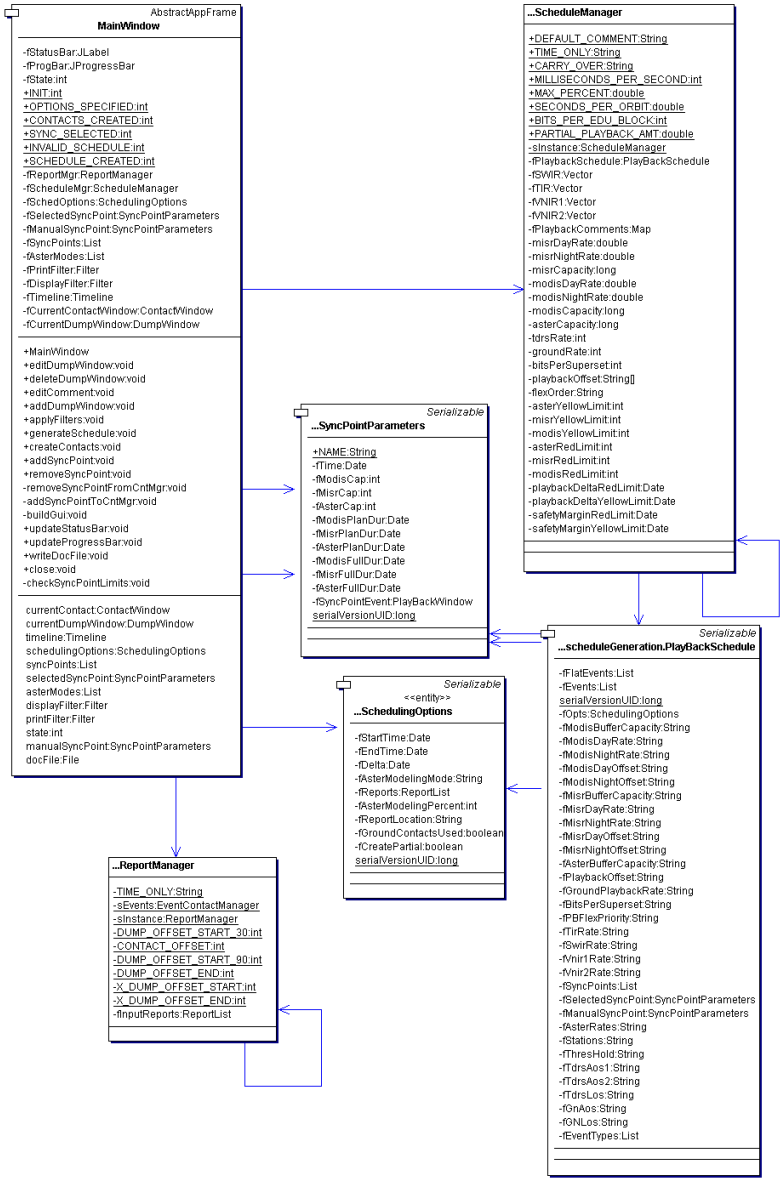
The following sequence diagram describes the interactions among classes when the user selects local as the report location and manually specifies the names and locations of input reports on the local machine.



6.5 Main Window

The MainWindow is the workhorse class for the GUI thread. Through this class, the user controls schedule generation, enters scheduling parameters, changes modeling parameters, and prints/saves generated playback schedules.

6.5.1 Main Window Class Diagram



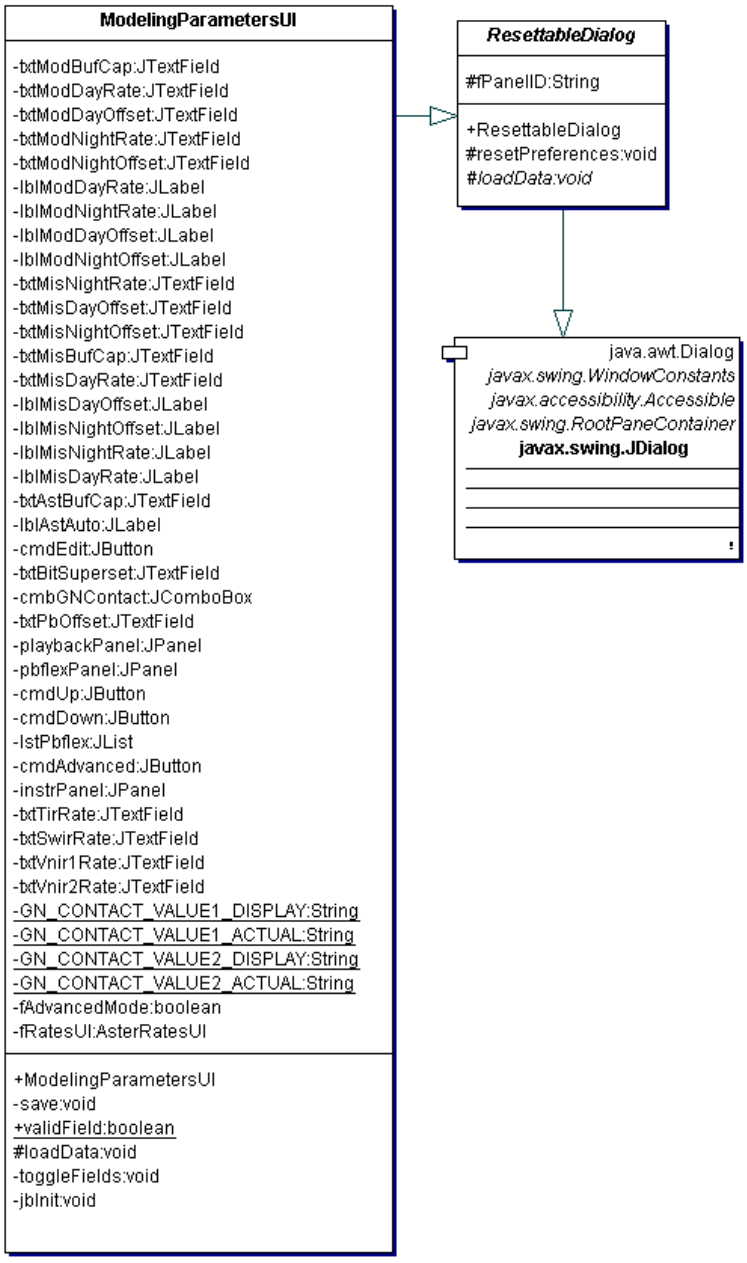
6.5.1.1 Major Methods

- ***public void addDumpWindow(DumpWindow dump)*** – adds the specified DumpWindow to the current contact.
- ***public void addSyncPoint(SyncPointParamaters sync)*** – adds a candidate sync point to the list of sync point events.
- ***public void applyFilters()*** – applies the filter settings to the timeline data.
- ***public void createContacts()*** – starts report parsing and event extraction.
- ***public void deleteDumpWindow(DumpWindow dwin)*** – remove the specified DumpWindow from the current contact.
- ***public void editDumpWindow()*** – edit the selected dump window contents.
- ***public void generateSchedule()*** – starts the schedule generation process.

6.6 Parameter UI

The ParameterUI is a dialog window through which the user specifies and/or modifies the modeling parameters settings used during playback schedule generation.

6.6.1 Parameter UI Class Diagram

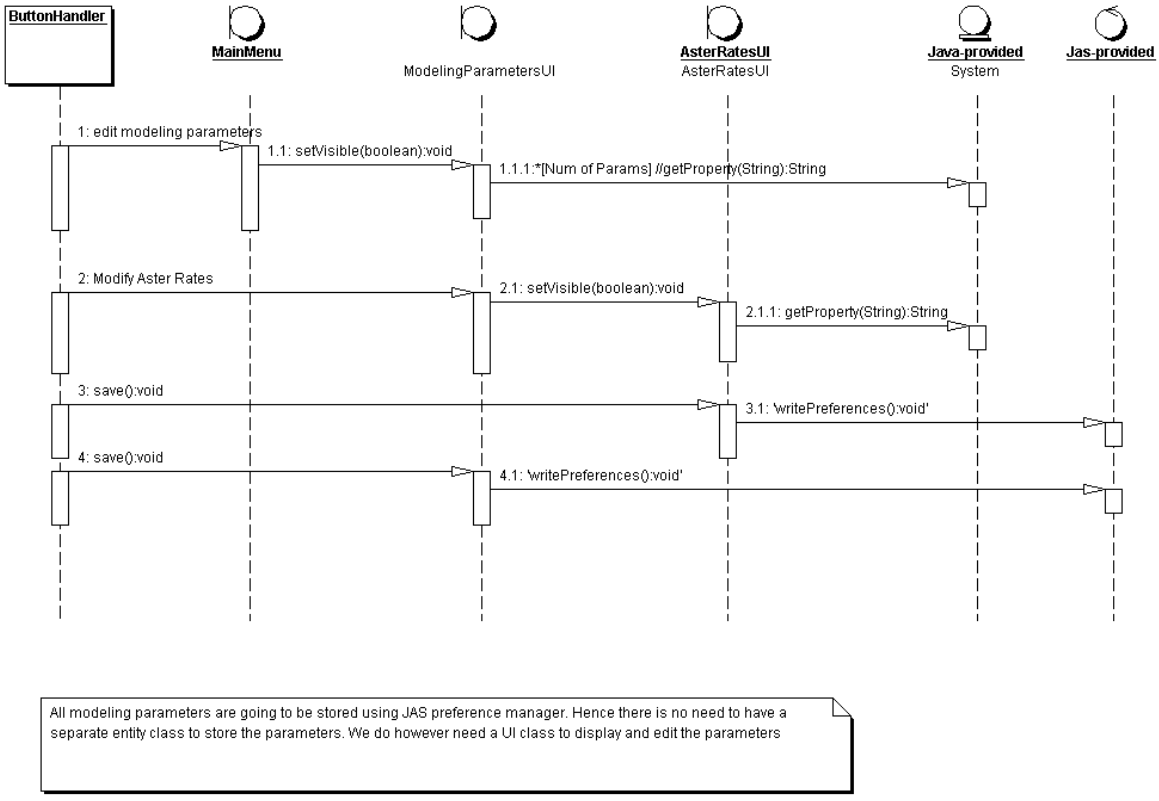


6.6.1.1 Major Methods

- ***public void save()*** – save the modified modeling parameters.
- ***public static Boolean validateField(JTextField field, String title, boolean checkFormat)*** – validates the data in the specified field.
- ***public void toggleFields()*** – hides or displays the advanced features of the Modeling Parameters UI.

6.6.2 Parameter UI Sequence Diagram

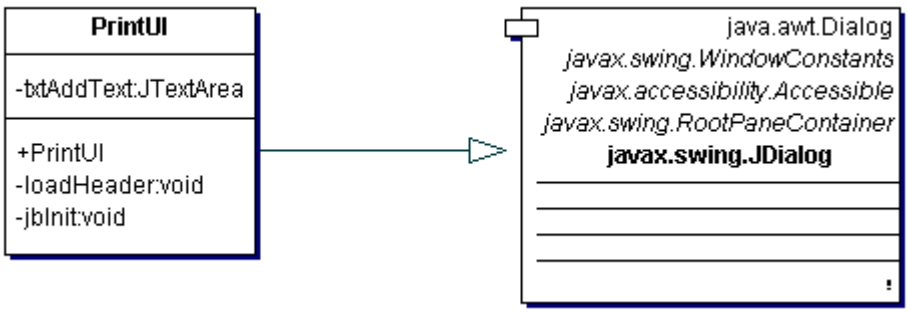
The following sequence diagram describes the interactions among classes when editing the modeling parameters.



6.7 Print Schedule UI

The PrintScheduleUI is a user interface through which the operator selects the printer or file to which the currently displayed schedule will be printed or piped.

6.7.1 Print Schedule UI Class Diagram

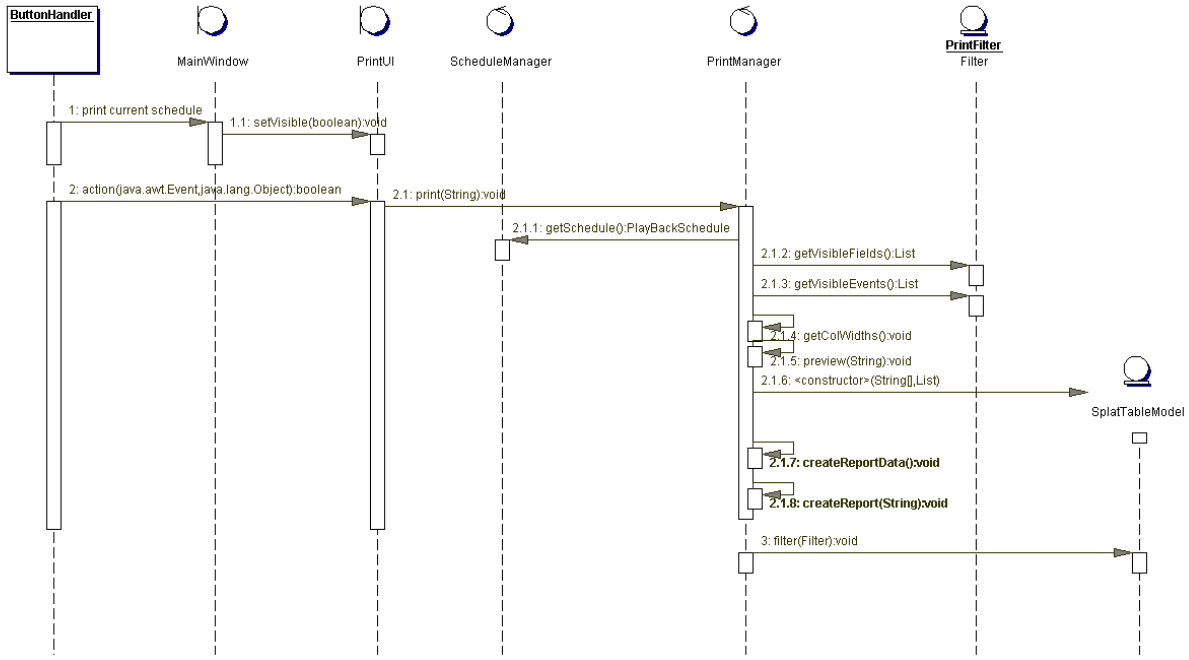


6.7.1.1 Major Methods

- ***public void loadHeader()*** – retrieves the template header information from a text file.

6.7.2 Print Schedule UI Sequence Diagram

The following sequence diagram describes the interactions among classes when printing a playback schedule.

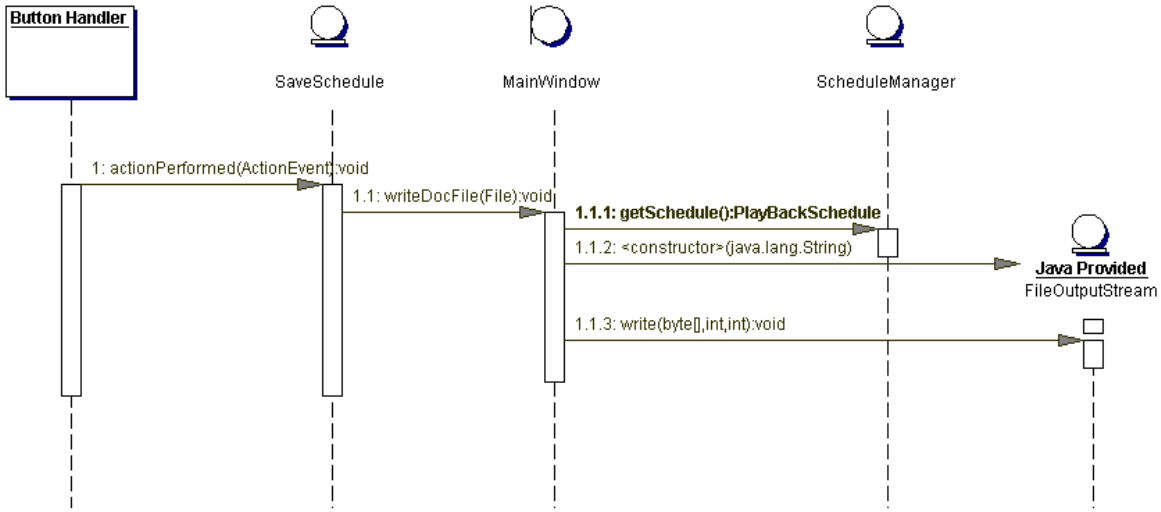


6.8 Save Schedule UI

The Save Schedule UI is implemented as a standard File Chooser Dialog (JFileChooser). It is displayed when the operator selects the save schedule option from the Main Menu. Through this UI the user selects the name and location for the file.

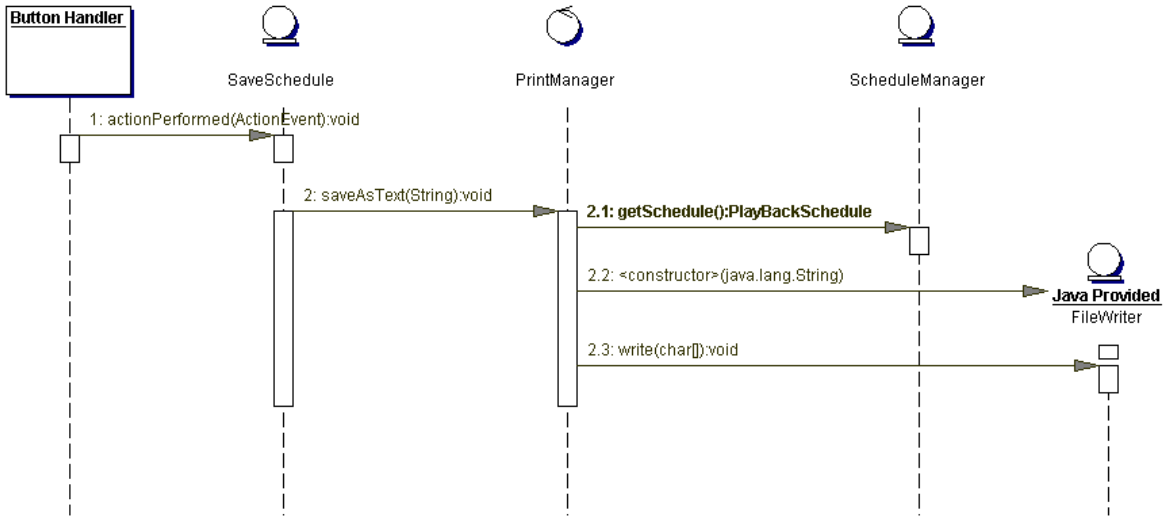
6.8.1 Save Schedule as Binary UI Sequence Diagram

The following sequence diagram describes the interactions among classes when saving a playback schedule to a binary file.



6.8.2 Save Schedule as Text UI Sequence Diagram

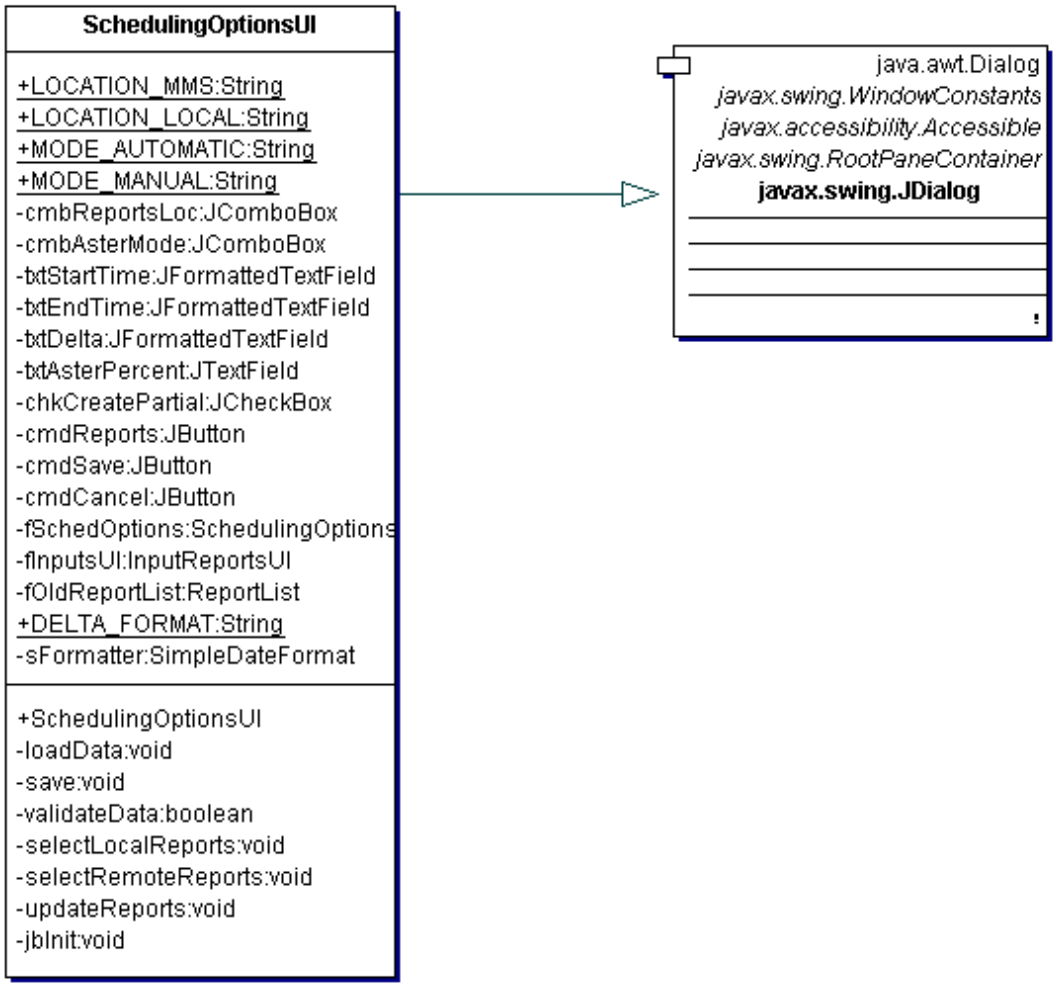
The following sequence diagram describes the interactions among classes when saving a playback schedule to a text file.



6.9 Scheduling Options UI

The Scheduling Options UI is implemented as a JDialog. Through this UI, the user enters and/or modifies the scheduling options. They can set the scheduling window, choose the location and name of the input reports, select the ASTER modeling mode and select whether or not ground contacts are used for playbacks.

6.9.1 Scheduling Options Class Diagram

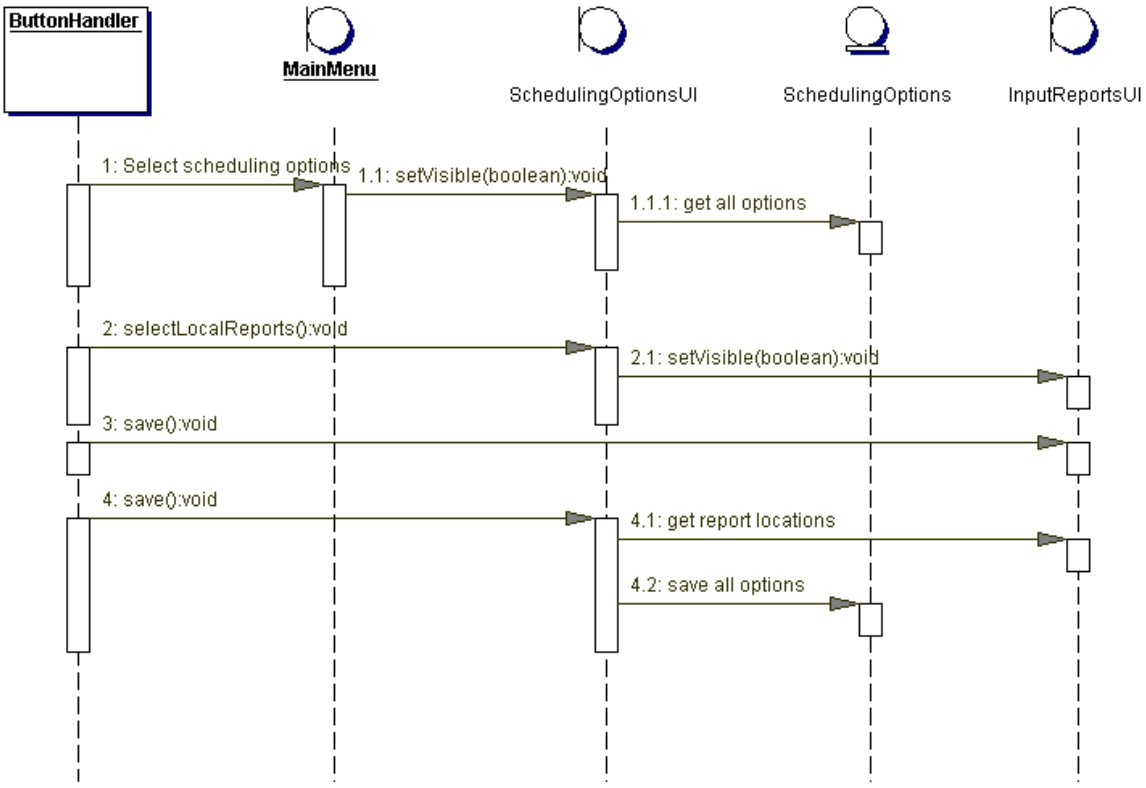


6.9.1.1 Major Methods

- ***public void selectLocalReports()*** – select reports locally on the current machine.
- ***public void selectRemoteReports()*** – coordinates remote report retrieval via the MMSReportRetriever.
- ***public void save()*** – save the user modified scheduling options.
- ***public boolean validateData(Boolean datesOnly)*** – validates the user provided values.

6.9.2 Scheduling Options Sequence Diagram

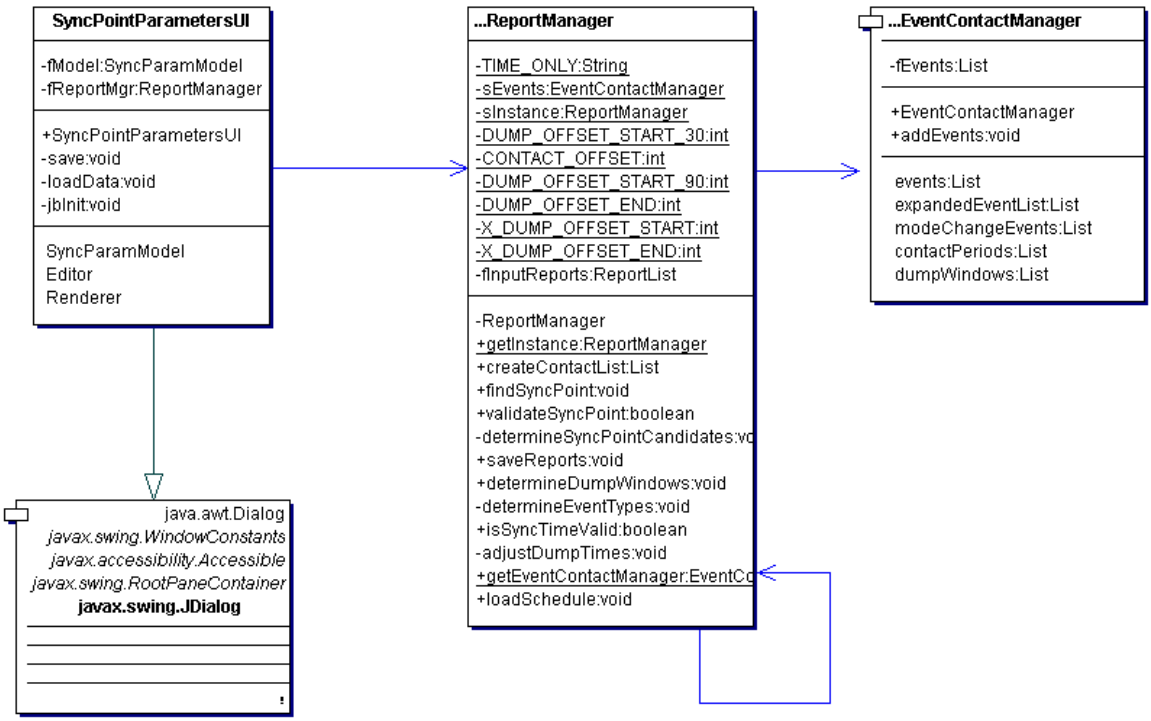
The following sequence diagram describes the interactions among classes when editing the scheduling option values.



6.10 Sync Point Parameters UI

The Sync Point Parameters UI provides a dialog through which the operator can select a synchronization point. All sync point candidate entries are displayed in this dialog. The current sync point will have the radio button next to the entry selected. The user can change the sync point by clicking on the radio button next to the desired entry.

6.10.1 Sync Point Parameters UI Class Diagram

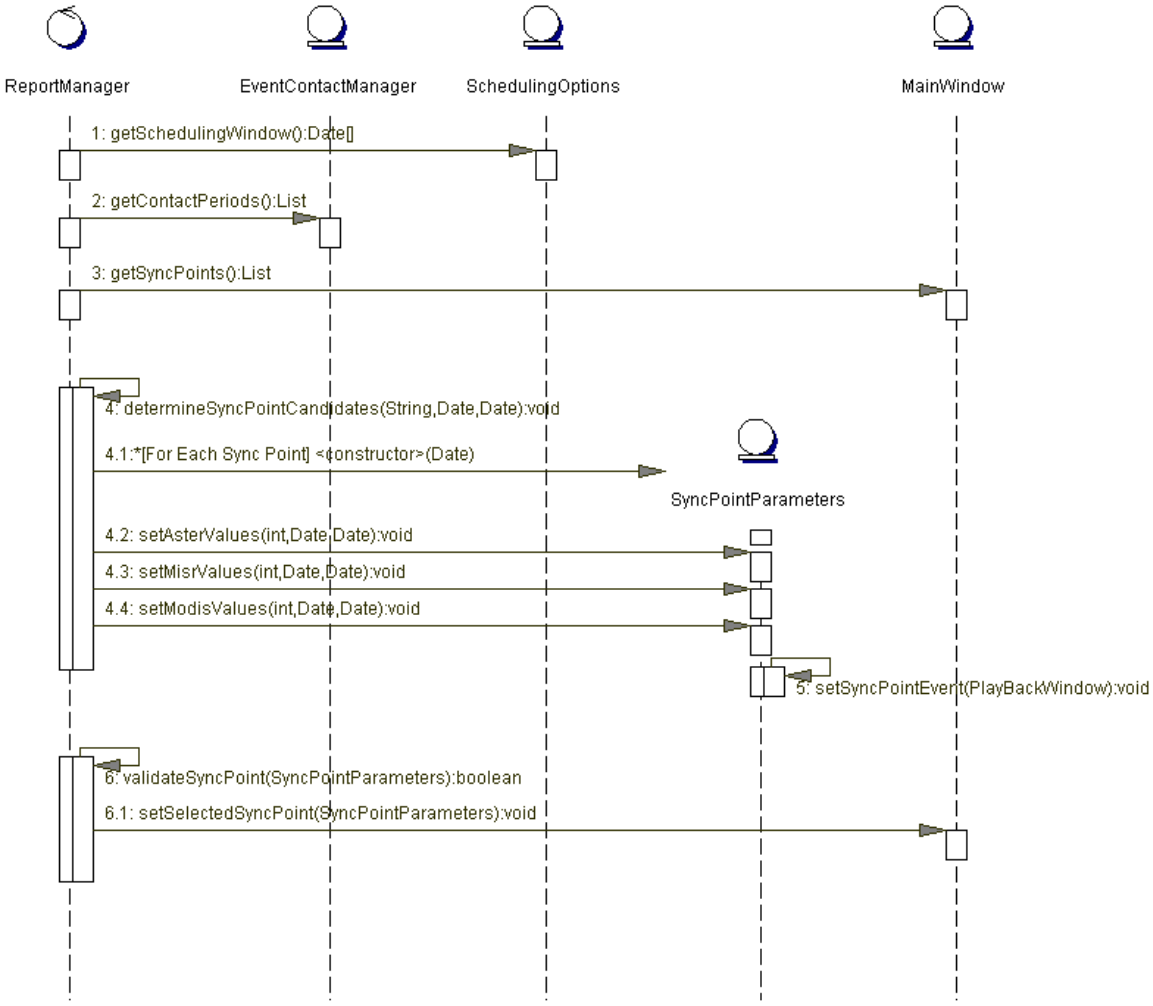


6.10.1.1 Major Methods

- ***public void save()*** – saves the modified sync point parameter values.

6.10.2 Sync Point Parameters UI Sequence Diagram

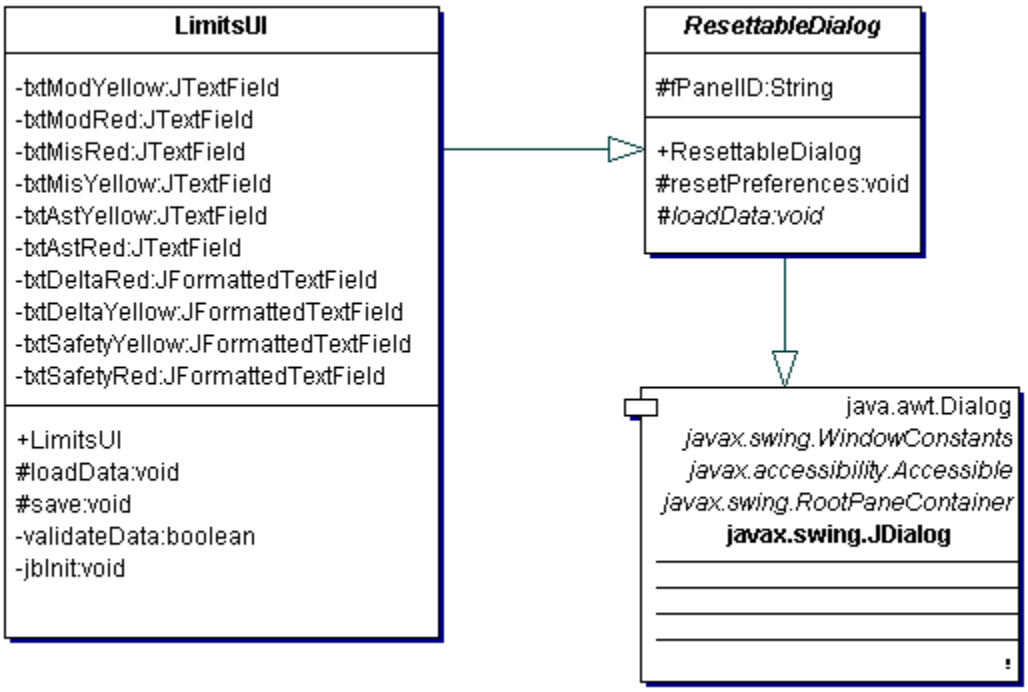
The following sequence diagram describes the interactions among classes when editing the synchronization point parameters.



6.11 Red & Yellow Limits UI

The Red & Yellow Limits UI is a dialog through which the user edits the threshold values for checking and tagging buffer usages, and playback start and stop times as critical or suspect. These values represent suspect and critical limits set by the operator and allow the tool to color code playback values that exceed the specified thresholds.

6.11.1 Red & Yellow Limits UI Class Diagram

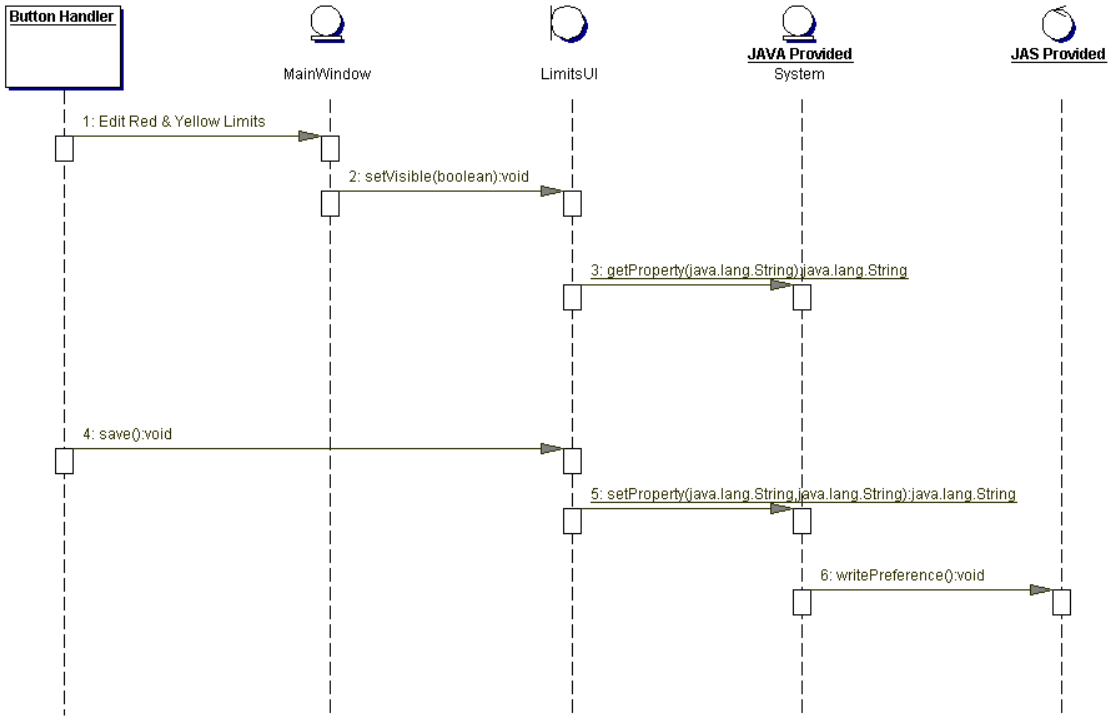


6.11.1.1 Major Methods

- ***public void save()*** – saves the modified Red & Yellow parameter values.

6.11.2 Red & Yellow UI Sequence Diagram

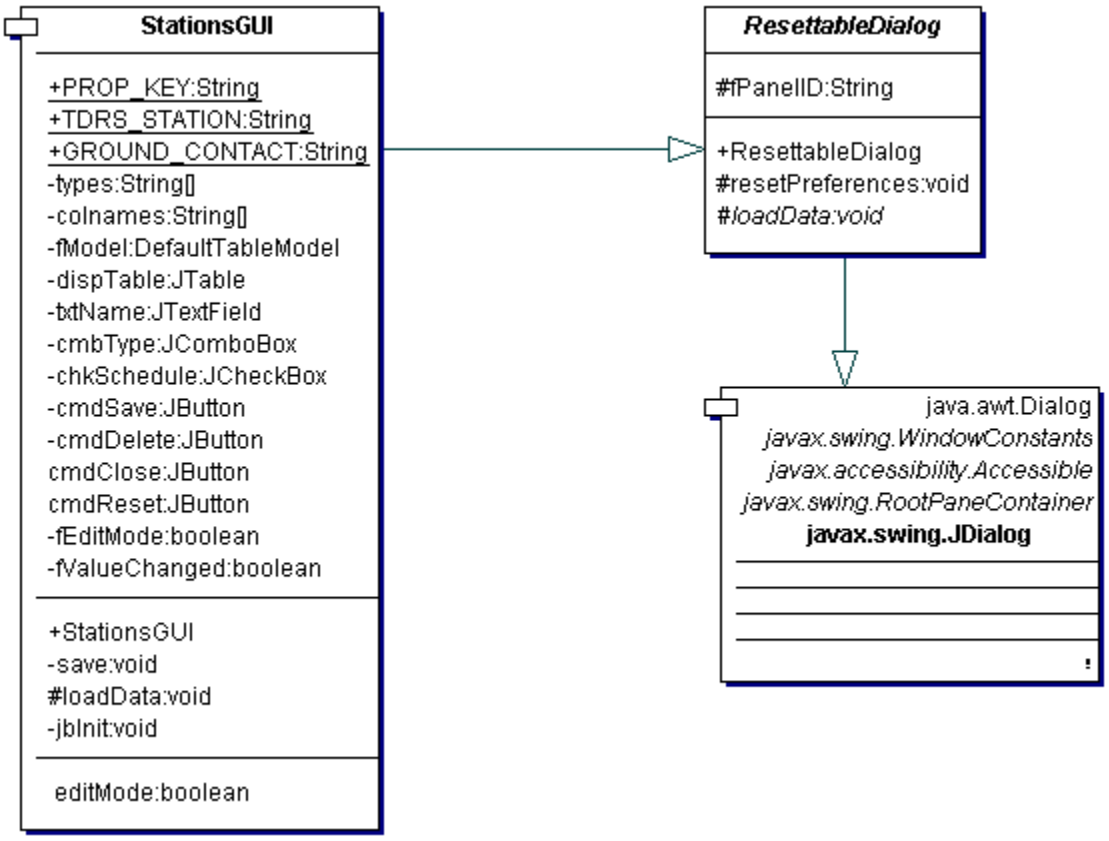
The following sequence diagram describes the interactions among classes when editing the Red & Yellow limit settings.



6.12 Station Management UI

The Station Management UI is dialog through which the user edits the list of available TDRS and WOTIS contacts. Each entry contains a type of contact (TDRS or GN), the name of the contact, and flag indicating whether or not ASSET should schedule playbacks in the contact type.

6.12.1 Station Management UI Class Diagram

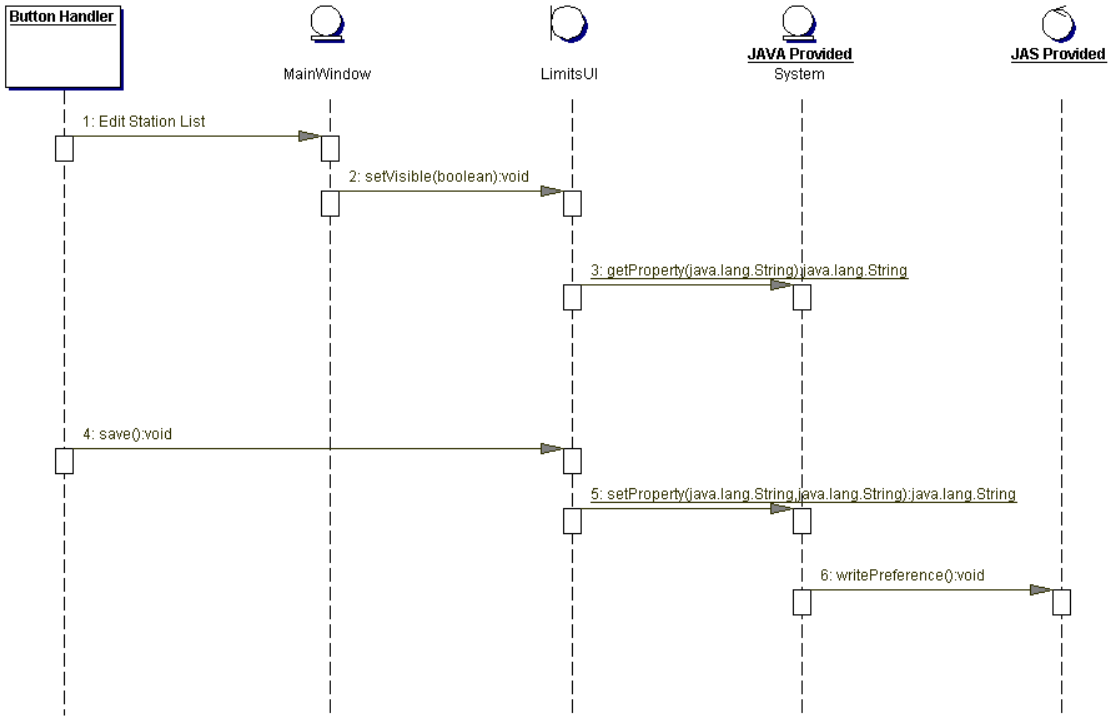


6.12.1.1 Major Methods

- ***public void save()*** – saves the modified stations management parameter values.

6.12.2 Station Management UI Sequence Diagram

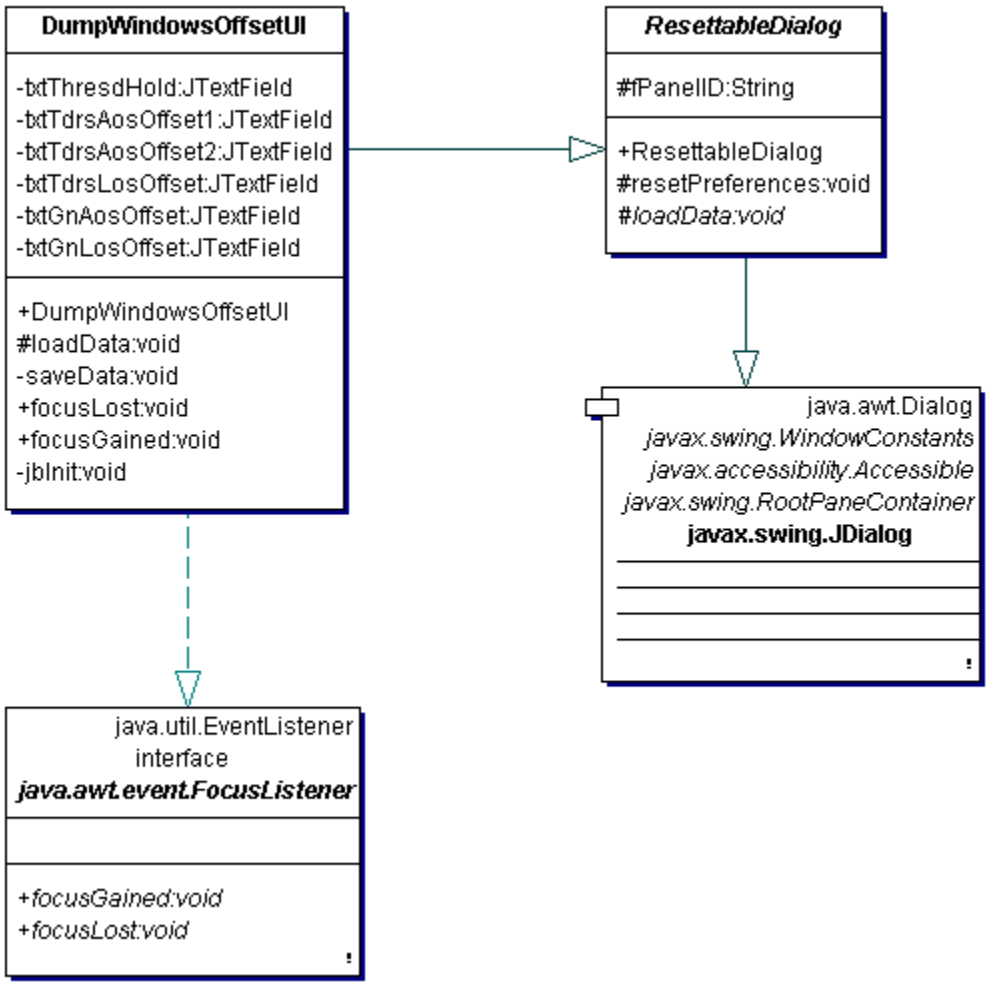
The following sequence diagram describes the interactions among classes when editing the list of TDRS and GN contacts.



6.13 Dump Window Offsets UI

The Dump Window Offsets UI is dialog through which the user edits the offsets used during contact creation to determine a dump windows location with respect to it's associated contact window.

6.13.1 Dump Window Offsets UI Class Diagram

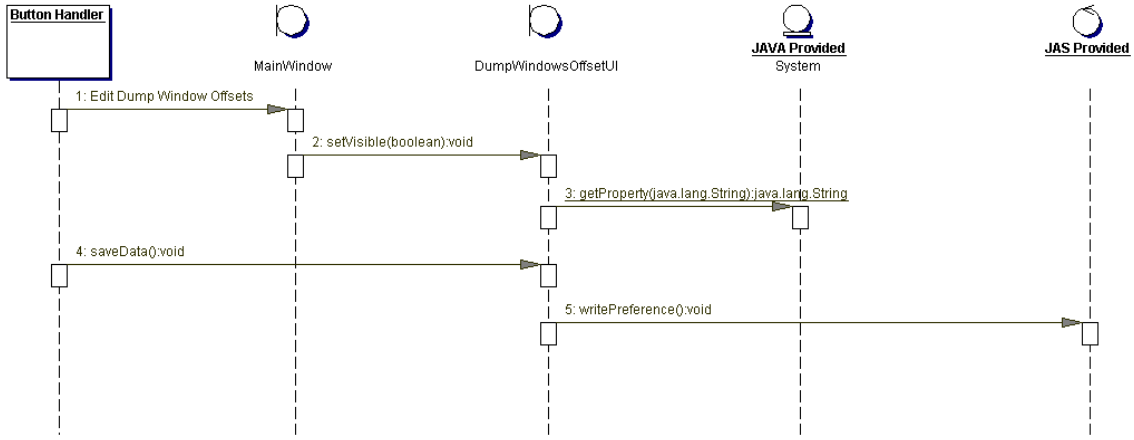


6.13.1.1 Major Methods

- ***public void save()*** – saves the modified dump window offset parameter values.

6.13.2 Dump Window Offsets UI Sequence Diagram

The following sequence diagram describes the interactions among classes when editing the dump window offsets.



7 Process View

This section presents an architectural view that describes the concurrent aspect of the system: tasks (processes and threads), persistent objects and their interactions.

7.1 Processes and Threads

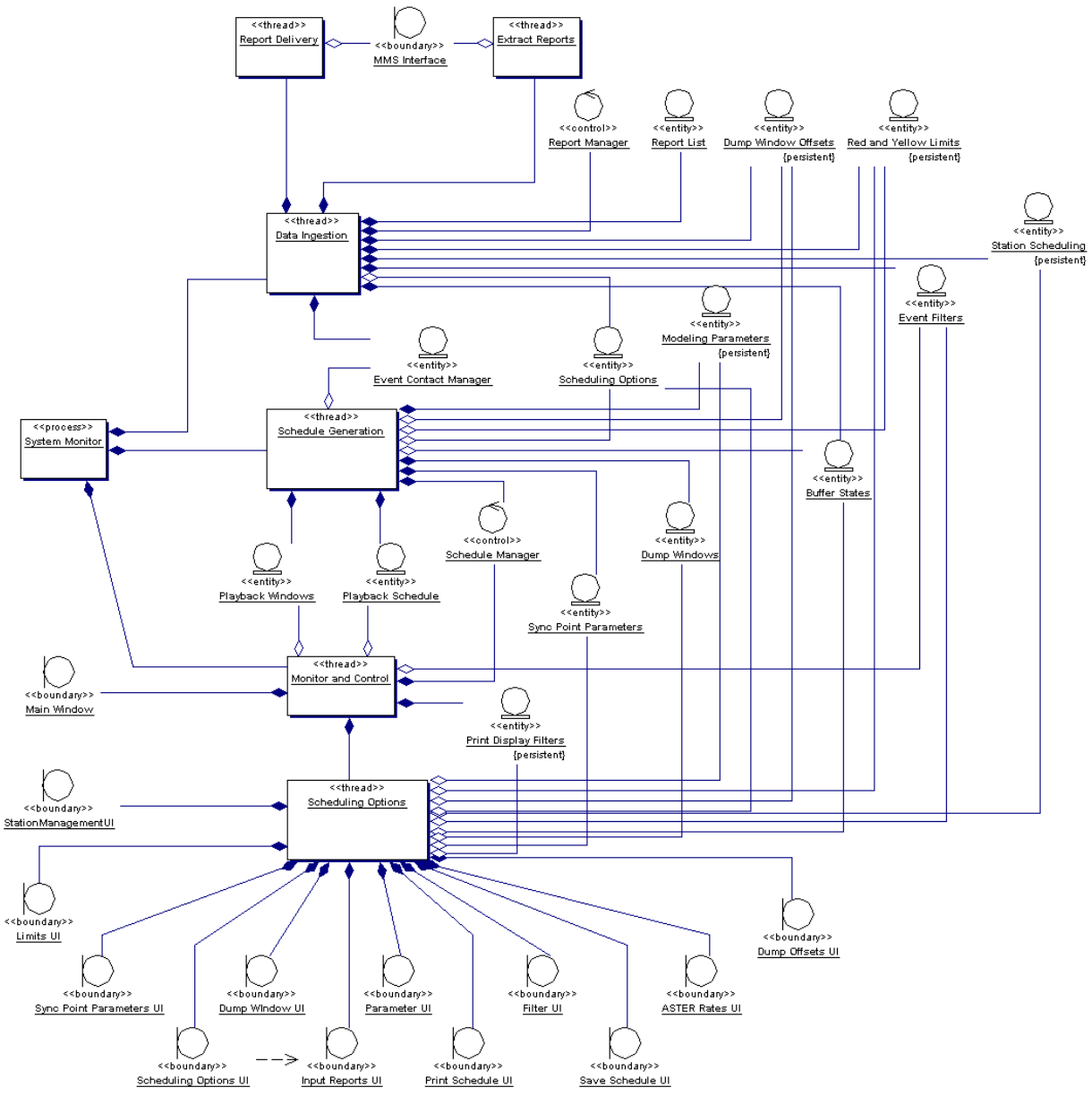


Figure 7-1: ASSET Process View Diagram

8 Size and Performance

This section provides a description of the major dimensioning characteristics of the software that affect the architecture, as well as the target performance constraints.

1. **Average schedule generation times** (see System Requirements Document) – These requirements were derived from discussions with the Flight Operations Team (FOT) staff charged with generating special event schedules. Scheduling these events manually can take anywhere from 30 minutes to several days depending on the complexity of the event. Since ASSET will automate the manual task of generating these schedules, a significant time savings is expected and schedule generation times would be in-line with those mentioned in the requirements specification.
2. **Average report extraction times** (see System Requirements Specification) – These requirements were derived from discussions with the FOT staff. Since the ASSET feature supporting automated extraction of reports adds only a minimal overhead over the manual procedure, the report extraction time limits are reasonable.
3. **Impact to MMS operation** – The ASSET Report Retrieval process must operate such that it does not degrade the performance of the MMS software. The features provided by the Report Retrieval process are similar to those performed by an operator at the MMS console to manually extract reports from the MMS database. The Report Retrieval process will add little overhead to the above operations and thus minimal or no impact on MMS performance is expected.

9 Quality

The software architecture described will support the reliability and supportability requirements for the system by virtue of the loose coupling between the MMS system and the ASSET tool and the modular design of the system. Usability requirements are addressed by using simple, yet functional, graphical user interfaces for operator interaction with the system.

10 Modeling Limitations/Constraints

The following constraints/limitations are placed on the model of the Terra SSR Buffers in the ASSET System.

10.1 Events

The following events are parsed from the input reports and used to determine playback schedules:

1. Nadir Term Crossing to Day Events
2. Nadir Term Crossing to Night Events
3. K-Band AOS Events
4. K-Band LOS Events
5. MA AOS Events
6. MA LOS Events
7. S-Band AOS Events
8. S-Band LOS Events
9. X-Band AOS Events
10. X-Band LOS Events
11. ASTER RTCS Events
12. SSR Buffer States

10.2 Constraints

The following constraints are used during schedule generation:

1. All specified rates are specified in bits/second (bps).
2. The ASTER IMAGING rate will be one of the following:
    a. User selectable percentage – default of zero
    b. Automated – parse ASTER RTCS events from ATC Load Report.
3. The following ASTER RTCS_IDs will be used for scheduling in automated ASTER mode:
    a. RTCS_ID = 5
        1. SWIR turned on 10 seconds after event.
        2. TIR turned on 9 seconds after event.
        3. VNIR1 turned on 11 seconds after event.
        4. VNIR2 turned on 12 seconds after event.
    b. RTCS_ID = 7
        1. TIR is turned off 4 seconds after event.
        2. SWIR is turned off 3 seconds after event.
        3. VNIR1 is turned off 2 seconds after event.
    c. RTCS_ID = 8

        1. TIR is turned on 7 seconds after event.
        2. SWIR is turned on 8 seconds after event.
    d. RTCS_ID = 9
        1. TIR is turned off 3 seconds after event.
        2. SWIR is turned off 2 seconds after event.
    e. RTCS_ID = 13
        1. VNIR 3 is turned off 13 seconds after event.
    f. RTCS_ID = 14
        1. TIR is turned on 2 seconds after event and turned off 32 seconds after event.
4. The imaging rates for the ASTER Instruments are as follows:
    a. VNIR1 – 31,019,000 bps
    b. VNIR2 – 31,019,000 bps
    c. SWIR – 23,053,00 bps.
    d. TIR – 4,109,000 bps
5. The MODIS Day Imaging rate is 10,686,117.6470588 bps.
6. The MODIS Night Imaging rate is 3,192,760.18099548 bps.
7. The MISR Day Imaging rate is 6,486,877.8280543 bps.
8. The MISR Night Imaging rate is 1,013.57466063348 bps.
9. MODIS transition to day mode is keyed to “Nadir Term Crossing To Day” events extracted from the Orbital Events report.
10. MODIS transition to night mode is keyed to “Nadir Term Crossing to Night” event extracted from the Orbital Events report.
11. MISR Transitions to Day 3.5 minutes before MODIS.
12. MISR Transitions to Night 3.5 minutes after MODIS.
13. The K band contact playback rate is 150,000,000 bps.
14. The X-band contact playback rate is 150,000,000 bps.
15. The ASTER Buffer capacity is 84,813,557,760 bits or 58 Supersets.
16. The MISR Buffer capacity is 29,246,054,400 bits or 20 Supersets.
17. The MODIS Buffer capacity is 52,642,897,920 bits or 36 Supersets.
18. The conversion rate for bits to supersets is 1,462,302,720 bits/Superset.
19. The offset between the start of a Dump Window and the start of a playback window will be use configurable with a default value of 0.
20. If an S-band contact starts more than 60 seconds earlier than it’s corresponding K-band contact then the Dump window start time is offset 30 seconds after the start of the k-band contact, and the dump window stop time is offset 180 seconds before the end of the K-band contact.
21. If the start time of a S-band contact doesn’t precede it’s corresponding X-band contact by more than 60 seconds, the dump window start time is offset from the start of the S-band contact by 90 seconds and the dump window stop time is offset by 180 seconds before the end of the K-band contact.

11 Glossary

Table 11-1 contains a list of the acronyms and abbreviations used in this document along with a brief description of each acronym.

Acronym/Abbreviation	Term	Definition
AOS	Acquisition of Signal	A term describing the acquisition of signal for a TDRS satellite or ground station. Used in determining dump times for Terra.
ASSET	Advanced Ssr SchEduling Tool	The new name for SPLAT.
ASTER	Advanced Spaceborne Thermal Emission and Reflection	Instrument on-board TERRA owned and operated by the Japanese space agency.
ATC	Absolute Time Command	
FOT	Flight Operations Team	The group of engineers charged with monitoring and maintaining a spacecraft on orbit.
FTP	File Transfer Protocol	A standard protocol for transferring files across networks of computers.
GN	Ground Network	Refers to the series of ground stations (Alaska and Norway) to which SSR buffer data may be downlinked.
GOC	Goal Oriented Commanding	The predecessor to SPLAT. GOC was to provide a system that allowed an operator to command a satellite or constellation of satellites using natural language commands and goals.
GSFC	Goddard Space Flight Center	
GUI	Graphical User Interface	A graphical interface (dialogs, etc.) through which a user interfaces (communicates) to a computer system or program.
JAS	Java Application Shell	A framework for building Java applications.
LAN	Local Area Network	
LOS	Loss of Signal	A term describing the loss of signal for a TDRS satellite or

		ground station. Used in determining dump times for Terra.
MISR	Multi-angle Imaging Spectro-Radiometer	An instrument on the Terra spacecraft
MODIS	Moderate Resolution Imaging Spectrometer	An instrument on the Terra spacecraft.
MMS	Mission Management Software	Unique to EOS, this system is the primary mission planning system for Terra. Among other products, it creates the TDRS Contact Report, and includes basic models for generating command loads.
NASA	National Aeronautics and Space Administration	
RTCS	Real Time Command Sequence	
SPLAT	SSR Playback Automation Tool	The tool being developed to assist with SSR buffer playback scheduling for special events.
SSR	Solid State Recorder	This is Terra's on-board storage device. It operates using buffers wherein data from each instrument (4 buffers total) and housekeeping data are stored for later downlink to a ground station.
SWIR	Short Wave Infrared	An ASTER instrument.
TDRS	Tracking and Data Relay Satellite	A satellite to which the SSR buffer information may be relayed.
TIR	Thermal Infrared	An ASTER instrument
UI	User Interface	Synonymous with GUI.
UML	Unified Modeling Language	
VNIR	Visible and Near Infrared	An ASTER instrument